2

Self-Organizing Deformable Model: A Method for Projecting a 3D Object Mesh Model onto a Target Surface

Ken'ichi Morooka[1] and Hiroshi Nagahashi[2]
[1]*Graduate School of Information Science and Electrical Engineering, Kyushu University*
[2]*Imaging Science and Engineering Laboratory, Tokyo Institute of Technology*
Japan

1. Introduction

3D model fitting is one of the fundamental and powerful techniques in computer vision and computer graphics. The reason for using the fitting techniques includes establishing a relationship between an object of complex shape and a primitive object with a simple shape. Since object models often contain a huge number of points on the object surface, using the original models directly is computationally very expensive. On the other hand, the primitive models either have a compact data size or are represented by parametric functions. The relationship between the original and primitive models enables the complex shaped original models to be edited efficiently and easily through their corresponding primitives.

In addition, the fitting techniques provide the correspondence between multiple models by representing all models with a unified primitive model. This correspondence enables us not only to align the models but also to transfer the properties of one model, such as texture and motion data, to another model. Therefore, the fitting method is useful in a wide range of applications including Eigenspace-based object modeling (Allen et al., 2003; Blanz et al., 1998; Kurazume et al., 2007), texture mapping, 3D object morphing (Blanz et al., 1998) and computer animation (Noh et al., 2001; Sumner et al., 2004).

The fitting method has to meet certain requirements as described below. Appropriate primitive models are determined depending on the shape of target objects and/or the purpose of using the primitives. The choice of primitive models has an influence on some factors including the accuracy of recovering the object and the mapping distortion. Hence, the fitting method needs to deal with an arbitrary surface as a primitive. Some applications use the correspondence between given models. This correspondence can easily be found by fitting the models to a common primitive. Then, a fitting method is desired that allows users to specify the mapping between features of the models, such as mapping eyes to eyes and so on. Few existing methods, however, consider both these requirements simultaneously.

This paper presents a new method for fitting a deformable model to the points of a target object (Morooka et al., 2005; 2007). The deformable model is called a Self-organizing Deformable Model (SDM). The proposed fitting method is composed of two major steps.

First, we generate a rough model of the object by deforming the SDM based on competitive learning. Next, the accuracy of the model is improved by minimizing an energy function. The framework of the SDM allows users to move some vertices included in the SDM towards specific points on the target surface, and to collect the vertices in particular regions on the surface. This control of the SDM deformation process provides preferable models depending on the applications using them.

1.1 Previous and related works

Many researchers have concentrated on model fitting techniques. One fitting method uses a mapping function(Eck et al., 1995; Floater et al., 2003), which enables us to project the object mesh model onto a primitive object surface such as a planar or spherical one. There are several types of mapping functions including harmonic mapping and conformal ones. Most of the traditional methods that use a mapping function have been designed only for their specific primitives. Because of this limitation, the methods are harder to generalize. In (Tarini et al., 2004), although, a set of cubes is used as the primitive, the combination of several kinds of primitives would be more appropriate for the compact low-distortion mapping of complex shapes.

Deformable models have the potential for achieving such mapping by regarding the set of multiple primitives as the deformable model. This is because the deformable model can adapt to the new shape by deforming the original shape. The approaches using deformable models can be categorized into the deformation methods with or without a certain number of constraints. The constraints are the preservation of the volume and/or the shape features of the model. Constrained deformable models can generate a continuous global relationship between the initial and the deformed models. The relationship is used to create model animations, and transfer deformation from one model to another (Noh et al., 2001; Sumner et al., 2004). One technique for the constrained deformation is to modify the model by deforming its embedded space or the underlying skeleton of the model (Ju et al., 2005; Kavan et al., 2005; Sederberg et al., 1986). Recent approaches (Kun et al., 2004; Lipman et al., 2005; Zayer et al., 2005) introduce the Laplace or Poisson equations, and compute the vertex displacements which describe the difference between the initial and the deformed models. The deformable model without the constraints applies where the relationship between the initial and the deformed models is an arbitrary deformation. Such deformable models are useful for mapping between two models with different shapes. Therefore, we focus on the the unconstrained deformable models, and these types of the models are simply called deformable models.

In the method using deformable models(Duan et al., 1998; Gibson et al., 1997; Lachaud et al., 1999; McInerney et al., 2000), a mesh of a plane or sphere is often used as the initial deformable model, and then the mesh model is deformed by moving its vertices to fit the target object. Generally, fitting the deformable model to a target object is formulated as a minimization problem of an energy function, which measures the deviation of the model from the target. Since the model deformation uses both vertices of the model and numerous points on the target surface, the optimum deformation must be found in a large search space that includes many local minima. In obtaining such a deformation, certain initial conditions of the deformable model have a great influence on the computational cost and the fitting accuracy of

the resulting model. However, the estimation of the initial model becomes more difficult when the shape of the object is complicated. It is not so easy for users to implement the preferable deformation. For example, an inadequate deformation may lead to certain problems such as the self-intersection of the model. Therefore, traditional methods that use deformable models need to find the optimal movement of vertices through trial-and-error.

Recent research (Kraevoy et al., 2004; Schreiner et al., 2004) has reported methods for finding the relationship between two models that preserves the correspondence of specific features. Although these methods obtain good mappings, they lead to high computational complexity and/or an increase in the data size of the output model compared with that of the original one.

The basis of the SDM deformation is the combination of competitive learning and an energy minimization method. In much the same way, some methods (Barhak et al., 2001; Ivrissimtzis et al., 2004; 2003; Liou et al., 2005; Yu, 1999) utilize a framework of competitive learning to recover the whole object surface. An exception to these methods is that of (Barhak et al., 2001), which focuses on a range image. The original purpose of competitive learning is to obtain a network that represents the distribution of given points. Therefore, direct use of competitive learning causes the accuracy of the resulting model to be inadequate for approximating the original object surface; that is, the model has a blurred surface of the original target object (Liou et al., 2005). Some methods (Ivrissimtzis et al., 2004; 2003; Yu, 1999) introduce a process of changing the number of vertices and their connectivity during the deformation. Contrastingly, in our proposed method, the SDM can recover a target surface while retaining the architecture of the SDM. This characteristic enables us to represent objects with a unified form, and to easily establish a correspondence between multiple models. Our method can, therefore, be used in various applications in computer vision and computer graphics.

2. Self-organizing deformable model

2.1 Notations

The SDM is a deformable mesh model represented by triangular patches. According to the notation adopted in (Lee et al., 1998), the SDM $\mathcal{M}$ can be regarded as a two-tuple:

$$\mathcal{M} = (\mathcal{V}, \mathcal{K}), \tag{1}$$

where $\mathcal{V}$ is the set of vertices of the SDM. The SDM contains N_v vertices, with each vertex $\boldsymbol{v}_i$ $(1 \leq i \leq N_v)$ of the SDM having 3D positional information. An abstract simplicical complex, $\mathcal{K}$, contains all the adjacency information for $\mathcal{M}$. In other words, $\mathcal{K}$ includes three different sets of simplices: a set of which includes uniquely identified vertices $\{i\} \in \mathcal{K}_v$, a set of edges $e = \{i, j\} \in \mathcal{K}_e$, and a set of faces $f = \{i, j, k\} \in \mathcal{K}_f$. Therefore, $\mathcal{K} = \mathcal{K}_v \cup \mathcal{K}_e \cup \mathcal{K}_f$. Our edge representation has a commutative property. For example, the edge e_3 as shown in Fig. 1 is composed of two vertices $\boldsymbol{v}_{i3}$ and $\boldsymbol{v}_{i4}$, and is described by $e_3 = \{i_3, i_4\} = \{i_4, i_3\}$. We represent each face of the SDM by a list of its three vertices arranged in a counterclockwise direction, e.g., the face f_2 shown in Fig. 1 is described by $f_2 = \{i_2, i_4, i_3\} = \{i_4, i_3, i_2\} = \{i_3, i_2, i_4\}$.

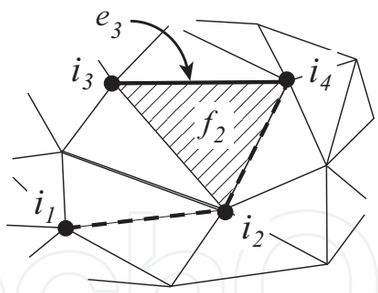


Fig. 1. Example of notations used in our SDM. Adapted from the article (Morooka et al., 2007). Copyright(c)2007 IEICE.

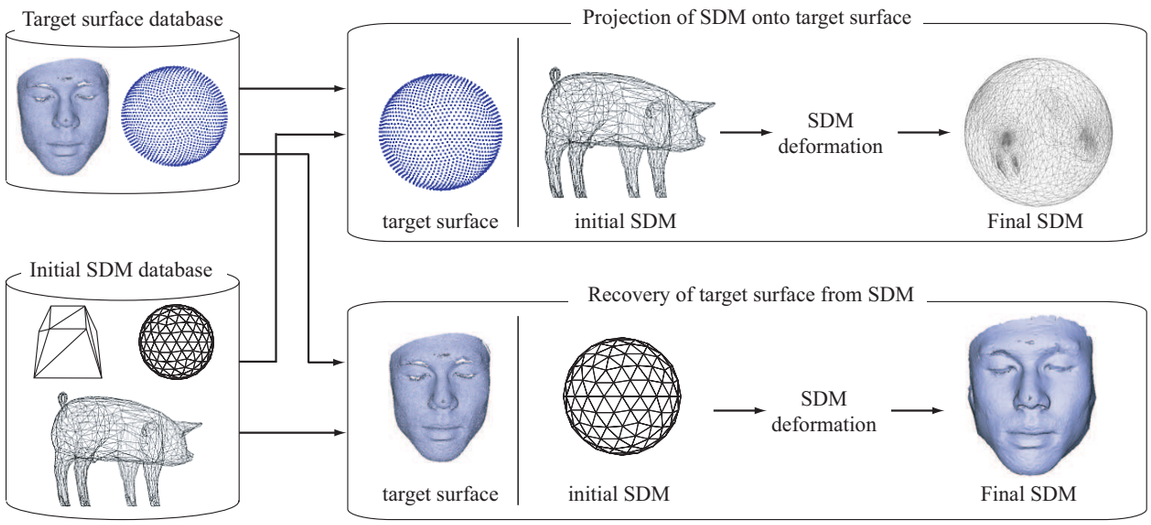


Fig. 2. Concept of Self-organizing Deformable Model. Adapted from the article (Morooka et al., 2007). Copyright(c)2007 IEICE.

Given two vertices $\boldsymbol{v}_{i1}$ and $\boldsymbol{v}_{i4}$ as shown in Fig. 1, a path from $\boldsymbol{v}_{i1}$ to $\boldsymbol{v}_{i4}$ is defined as a series of edges. Then, a topological distance $L(i_1, i_4)$ between the vertices is defined as the number of edges comprising the shortest path from $\boldsymbol{v}_{i1}$ to $\boldsymbol{v}_{i4}$. Another topological distance $L(i_4, i_1)$ from $\boldsymbol{v}_{i4}$ to $\boldsymbol{v}_{i1}$ is equal to $L(i_1, i_4)$. The shortest path from $\boldsymbol{v}_{i1}$ to $\boldsymbol{v}_{i4}$ is illustrated by dashed lines, and the topological distance $L(i_1, i_4)$ is 2. When the topological distance from a vertex $\boldsymbol{v}_i$ to an arbitrary vertex is r, the latter vertex is called the r-neighbors of $\boldsymbol{v}_i$ Given an arbitrary vertex $\boldsymbol{v}$, we regard as its neighbors the vertices for which the topological distance from each of the vertices to $\boldsymbol{v}$ is less than a threshold θ_r. Here, the maximum topological distance depends on the data size of the SDM, so we manually determine the value of the parameter θ_r according to the size of the SDM.

A target surface is represented by a set of points on the surface, called control points. We make the following three assumptions: 1) the surface orientation and surface normal at each control point are known; 2) the size of an initial SDM is sufficient to cover the target surface

completely; and 3) the SDM and the target surface have the same topological type. This paper focuses on object surfaces of genus 0. From these assumptions, we can choose arbitrary shapes for both the SDM and target surface. As shown in Fig. 2, the initial SDM and the target surface can be selected from various kinds of models besides spheres, animals and human faces. A mesh model selected as an initial SDM can be projected onto various kinds of target surface as in other methods for surface parameterization. Furthermore, like general deformable models, the SDM can reconstruct surfaces of target objects.

2.2 Formulation of the SDM deformation

There are two purposes for deforming the SDM. Considering the SDM notation in Eq.(1), the first purpose is to project the SDM onto the target surface by changing the vertex positions in $\boldsymbol{\mathcal{V}}$ while keeping the original topology for the vertices represented by $\boldsymbol{\mathcal{K}}$. This projection is called topology-preserving mapping. With an initial SDM $\boldsymbol{\mathcal{M}}^s$ and a final SDM $\boldsymbol{\mathcal{M}}^d$, the topology-preserving mapping Φ is formulated as

$$\Phi : \boldsymbol{\mathcal{M}}^s = (\boldsymbol{\mathcal{V}}^s, \boldsymbol{\mathcal{K}}) \longmapsto \boldsymbol{\mathcal{M}}^d = (\boldsymbol{\mathcal{V}}^d, \boldsymbol{\mathcal{K}}) \tag{2}$$

The second purpose of the SDM deformation is to approximate the target surface $\boldsymbol{\mathcal{S}}$ by the final SDM $\boldsymbol{\mathcal{M}}^d$. Competitive learning is used to represent the distribution of control points on the target surface. From a geometrical point of view, when applying competitive learning to the SDM deformation, the resulting SDM tessellates the target surface into the Voronoi regions, each of which includes almost the same number of control points on the target surface. The SDM cannot always recover the complex shape. For example, if we fit a spherical surface model as an initial SDM to the female model 'Igea' as shown in Fig. 4(c), and the fitting does not meet the second purpose, the final SDM is blurred and incomplete as shown in Fig. 7(c). Therefore, we formulate an error function for the approximated model to recover the target surface with good accuracy.

The second purpose is formulated by an error function for the approximated model. Various definitions of error functions are given in (Cignoni et al., 1998), and the error function in our method is defined as follows. Given a control point $\boldsymbol{p}$ on $\boldsymbol{\mathcal{S}}$, we consider a line that starts at $\boldsymbol{p}$ and is perpendicular to the plane expanded by the face $f \in \boldsymbol{\mathcal{K}}_f$. If the line intersects inside the face f, we let the intersection condition be true and calculate the distance between $\boldsymbol{p}$ and f. From the faces that satisfy the intersection condition, the face with the minimum distance is regarded as the corresponding face of $\boldsymbol{p}$. After finding all the correspondences between faces and control points, for each face f_m, its corresponding control points are collected into a set $\boldsymbol{\Omega}_m$. Here, we denote as $\boldsymbol{\Gamma}_i$ the set of faces $f_m \in \boldsymbol{\mathcal{K}}_f$ that include the vertex $\boldsymbol{v}_i$. Then, the error function is defined as the distance $D(\boldsymbol{\mathcal{M}}^d, \mathcal{S})$ between $\boldsymbol{\mathcal{M}}^d$ and $\mathcal{S}$:

$$D(\boldsymbol{\mathcal{M}}^d, \mathcal{S}) = \sum_{\{i\} \in \boldsymbol{\mathcal{K}}_v} \sum_{f_m \in \boldsymbol{\Gamma}_i} \sum_{\boldsymbol{p}_u \in \boldsymbol{\Omega}_m} \frac{\{H(\boldsymbol{p}_u, f_m)\}^2}{3|\boldsymbol{\Gamma}_i||\boldsymbol{\Omega}_m|}, \tag{3}$$

where $|\boldsymbol{\Gamma}_i|$ and $|\boldsymbol{\Omega}_m|$ represent the total number of elements in $\boldsymbol{\Gamma}_i$ and $\boldsymbol{\Omega}_m$. The function $H(\boldsymbol{p}, f)$ returns the Euclidean distance between a control point $\boldsymbol{p}$ and the perpendicular foot of $\boldsymbol{p}$ on a face f.

From Eqs.(2) and (3), the deformation of $\mathcal{M}^s$ is formulated as the optimization problem of finding $\mathcal{M}^d$ that satisfies the following conditions:

$$\mathcal{M}^d = \tilde{\Phi}(\mathcal{M}^s); \tag{4}$$

$$\tilde{\Phi} = \arg\min_{\Phi} D(\Phi(\mathcal{M}^s), \mathcal{S}). \tag{5}$$

3. SDM deformation

3.1 Overview

Generally, the topology-preserving projection function Φ in Eq.(2) is nonlinear, and a deformation from $\mathcal{M}^s$ to $\mathcal{M}^d$ is not always unique. The Self-Organizing Map (SOM)(Kohonen et al., 2001) is a well-known method for finding a topology-preserving mapping, and it enables us to obtain a network that represents the distribution of the given input data. The network consists of a set of units and a set of edges connecting two units. Each unit has a positional vector that provides the position of the unit in the input data space. Initially, the SOM algorithm selects an input data randomly, and computes its Euclidean distance to every unit in the network. Next, the unit with the minimum distance is selected as the winner of the input data. The positions of the winner unit and its neighbor units are slightly adjusted towards the input data. These processes are repeated until no units can be further modified.

During the learning process of the SOM, the network is deformed by moving units in the input data space. We introduce the SOM framework into the SDM deformation by regarding the relationship between the SDM and control points as being equivalent to that of the SOM and the input data. Then, the vertices of the SDM and their 3D positions correspond to the units of the SOM and their positional vectors in the input data. However, from a geometrical point of view, problems occur when the SOM algorithm is applied directly to the SDM deformation. The major problems and their solutions are described below.

3.2 SDM deformation based on SOM

3.2.1 Determination of winner vertex

In the SOM algorithm, when a control point is selected randomly, a winner vertex is determined from the Euclidean distances between the control point and the vertices of the SDM. However, this determination may select a vertex that is on the opposite side of the target surface as shown in Fig. 3(a), where the control point and the selected vertex are marked by '●' and '■'. In this case, the selected vertex and its neighbors are moved into the target surface. As a result, the final SDM is projected onto the limited area of the target surface as shown in Fig. 3(b). Such a phenomenon is called a collapsed SDM.

To avoid this problem, traditional deformable models based on competitive learning (Ivrissimtzis et al., 2004; 2003; Yu, 1999) change the number of vertices of the models and their connectivity during the deformation. On the other hand, the purpose of the SDM deformation is to recover the target surface while keeping the architecture of the SDM. Considering this requirement, we deal with the collapse problem by using a signed distance SD between a

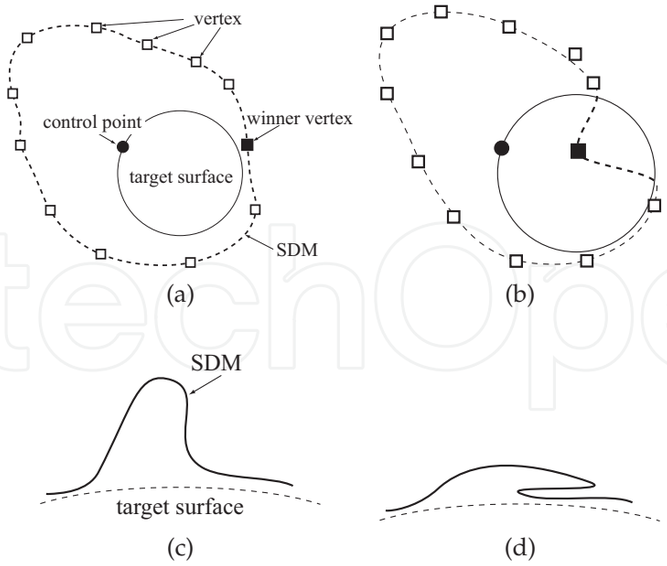


Fig. 3. Examples of a collapsed SDM caused by (a-b) moving certain vertices of the SDM into a target surface, and (c-d) folding over itself. Adapted from the article (Morooka et al., 2007). Copyright(c)2007 IEICE.

control point $\boldsymbol{p}$ and a vertex $\boldsymbol{v}$:

$$SD(\boldsymbol{v},\boldsymbol{p}) = \boldsymbol{n_p} \cdot (\boldsymbol{v} - \boldsymbol{p}), \tag{6}$$

where $\boldsymbol{n_p}$ is a unit normal vector at $\boldsymbol{p}$. SD has positive values if $\boldsymbol{v}$ is outside the surface on which $\boldsymbol{p}$ exists. Otherwise, SD takes negative values for $\boldsymbol{v}$ inside the surface. The absolute value of SD denotes the Euclidean distance between $\boldsymbol{p}$ and $\boldsymbol{v}$. In our method, the winner vertex is selected so that the signed distance between the vertex and the control point is non-negative and the minimum of all vertices.

3.2.2 Update of vertices

After determining the winner vertex $\boldsymbol{v}_c$ of a control point $\boldsymbol{p}$, $\boldsymbol{v}_c$ and its neighbors are updated by moving towards $\boldsymbol{p}$. In practice, the update of these vertices $\boldsymbol{v}_i$ is formulated as

$$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \epsilon(t)\lambda(i|c)\Delta\boldsymbol{v}_i; \tag{7}$$

$$\Delta\boldsymbol{v}_i = \boldsymbol{p} - \boldsymbol{v}_i, \tag{8}$$

where t is a time parameter, and $\epsilon(t)$ $(0 < \epsilon < 1)$ is the learning rate that determines the magnitude by which the vertex is adapted towards the point. At the start of implementing the SOM algorithm, $\epsilon(t)$ is usually set to a large value to achieve the global adjustment of the SDM. During the learning process, the learning rate gradually decreases to zero according to the specified learning rate function. We define $\epsilon(t)$ as follows (Fritzke et al., 1997):

$$\epsilon(t) = \epsilon_s(\frac{\epsilon_d}{\epsilon_s})^{t/T} \tag{9}$$

Here, ϵ_s and ϵ_d are the initial and the final learning rates.

The neighborhood function $\lambda(i|c)$ in Eq.(7) denotes the adaptation rate of an arbitrary vertex $\boldsymbol{v}_i$ for a winner vertex $\boldsymbol{v}_c$. In our method, $\lambda(i|c)$ is a decreasing function of the topological distance $L(i,c)$ between two vertices $\boldsymbol{v}_i$ and $\boldsymbol{v}_c$:

$$\lambda(i|c) = \exp[-\frac{1}{2}\{\frac{L(i,c)}{\sigma(t)}\}^2] \tag{10}$$

The standard deviation $\sigma(t)$ of the neighborhood function in Eq.(10) is called the neighborhood radius. At the start of the learning process, the large neighborhood radius is used to map the winner vertex and its neighbors onto the same portion of the target surface. Similar to the learning rate, the neighborhood radius typically decreases with time t, and the deformation algorithm converges. Therefore, the neighborhood radius is formulated by

$$\sigma(t) = \sigma_s(\frac{\sigma_d}{\sigma_s})^{t/T}, \tag{11}$$

where σ_s and σ_d are the initial and final values.

T is the maximum time for selecting winner vertices. The adaptation of the winner vertex and its neighbors causes the vertices to move towards the target surface, and may sometimes generate a collapsed SDM. To solve this problem, our method updates the vertices as follows.

For each vertex in the SDM, we compute the signed distances between the vertex and all control points, and select the closest control point with a non-negative signed distance. If the signed distance between the selected control point and the vertex is more than a given threshold τ, the vertex is assigned to a 'free' label. Otherwise, the vertex is assigned to an 'anchor' label. Here, to determine the value of the threshold τ, we use the average distance L between the 1-ring neighbor points of the target surface, and the threshold is set to $\tau = 2 * L$.

A vertex with a 'free' label is moved closer to the target surface while keeping its relative position with respect to the winner vertex $\boldsymbol{v}_c$. This movement is formulated by Eq.(7) and the translation vector $\Delta\boldsymbol{v}$ of the vertex $\boldsymbol{v}$

$$\Delta\boldsymbol{v}_i = \boldsymbol{p} - \boldsymbol{v}_i + \alpha(\boldsymbol{v}_i - \tilde{\boldsymbol{v}}_c) \tag{12}$$

where $\tilde{\boldsymbol{v}}_c$ is its 3D position before updating. The constant parameter α has to be small enough to avoid the self-intersection of the SDM. We empirically set α to $\alpha = 0.3$.

On the other hand, a vertex with an 'anchor' label is considered to be on the target surface. The adaptation of a vertex with the 'anchor' label using Eq.(12) sometimes causes the vertex to move into the target surface, and generate a collapsed SDM. To solve this problem, we move the vertex with an 'anchor' label along the target surface during the deformation. The movement is achieved by using a geodesic path from the vertex $\boldsymbol{v}$ to a control point $\boldsymbol{p}$. The function $\Psi(\gamma|\,\boldsymbol{v},\boldsymbol{p})$ $(0 \leq \gamma \leq 1)$ is defined so that the function returns the 3D positional vector of the point on the geodesic path. In addition, $\Psi(0|\,\boldsymbol{v},\boldsymbol{p}) = \boldsymbol{v}$ if $\gamma = 0$, and $\Psi(1|\,\boldsymbol{v},\boldsymbol{p}) = \boldsymbol{p}$ if $\gamma = 1$. Using the geodesic path, we update the vertex $\boldsymbol{v}_i$ with an 'anchor' label according to the following equation:

$$\boldsymbol{v}_i \leftarrow \Psi(\epsilon(t)\lambda(i|c)|\,\boldsymbol{v}_i,\boldsymbol{p}). \tag{13}$$

The geodesic path is computed efficiently using given techniques (Surazhsky et al., 2005). Then, all geodesic paths between two arbitrary points on the target surface are calculated off-line, and stored in a database. In the on-line SDM deformation, we obtain the geodesic path from the vertex merely by finding its closest point.

3.2.3 SDM after finishing the deformation

As stated above, our method introduces certain techniques to avoid a collapsed SDM. Nevertheless, the final SDM may contain foldovers, thus replicating the major problem of the original SOM algorithm. One solution to this problem is to restart the learning process using a different learning rate, the value of which decreases gradually, or by using another neighborhood function with a larger neighborhood radius (Duda et al., 2000). We can also use these ideas to create the SDM without foldovers.

For each vertex $\boldsymbol{v}$, we select the faces satisfying the following: 1) the topological distance between $\boldsymbol{v}$ and at least one of the vertices of the face is less than a given threshold; and 2) the perpendicular lines from $\boldsymbol{v}$ intersect the face. We compute the distance between $\boldsymbol{v}$ and each perpendicular foot of $\boldsymbol{v}$ on the selected face. If the minimum distance is less than a threshold, the SDM is considered to include a foldover, and we update the parameters of the learning rate and the neighborhood function in Eqs.(9) and (11):

$$\begin{aligned} \epsilon_s &\leftarrow h_1 \epsilon_s; \quad \epsilon_d \leftarrow h_2 \epsilon_d; \\ \sigma_s &\leftarrow h_3 \sigma_s; \quad \sigma_d \leftarrow h_4 \sigma_d. \end{aligned} \tag{14}$$

We empirically set $h_1 = 0.1$, $h_2 = 0.1$, $h_3 = 0.1$, and $h_4 = 0.1$. Using Eq.(14), the value of the updated learning rate decreases more slowly than the original learning rate. This changing of the learning rate can avoid a local minimum, that is, foldovers. On the other hand, the updated neighborhood function deals with only the vertices and the 1 or 2-ring neighbor to reduce the computational cost.

In addition, when applying the SOM algorithm to the SDM deformation, the final SDM $\boldsymbol{\mathcal{M}}^d$ has the characteristic that the Voronoi region of each vertex of the SDM includes almost the same number of control points. This means that $\boldsymbol{\mathcal{M}}^d$ cannot always recover the object surface completely because $\boldsymbol{\mathcal{M}}^d$ may not be satisfied by Eq.(5). We further deform the SDM by minimizing an energy function to improve the accuracy of the reconstructed target surface by the SDM.

The algorithm for the deformation chooses candidates for the next position of a vertex, and computes the energy function in the case of the vertex being moved to any of the candidates. We select as the next position of the vertex the candidate with the minimum valued energy function. Although the mesh deformation using the energy function is similar to the active balloon model, the energy function in our method is based on an external energy function, that represents the fitness of the SDM for the target surface. Using the error function in Eq.(3), the value of the energy function E for the vertex $\boldsymbol{v}_i$ is computed by

$$E(\boldsymbol{v}_i) = \sum_{f_m \in \boldsymbol{\Gamma}_i} \sum_{\boldsymbol{p}_u \in \boldsymbol{\Omega}_m} \{H(\boldsymbol{p}_u, f_m)\}^2. \tag{15}$$

3.3 Algorithm

1. Assign a 'free' label to all vertices.
2. Initialize the time parameter t to $t = 0$.
3. From the set of control points on the target surface, randomly choose a control point $\boldsymbol{p}^{(t)}$ at time t.
4. Determine the winner vertex $\boldsymbol{v}_c^{(t)}$ from
$$\boldsymbol{v}_c^{(t)} = \arg \min_{\{i\} \in \mathcal{K}_{\boldsymbol{v}}} SD(\boldsymbol{v}_i, \boldsymbol{p}^{(t)}). \tag{16}$$
5. Adapt the positions of the winner vertex $\boldsymbol{v}_c^{(t)}$ and its neighbors according to their labels:
$$\boldsymbol{v}_i \leftarrow \begin{cases} \boldsymbol{v}_i + \epsilon(t)\lambda(i|c^{(t)})\Delta\boldsymbol{v}_i & : (\boldsymbol{v}_i \text{ with a 'free' label}) \\ \Psi(\epsilon(t)\lambda(i|c^{(t)})|\ \boldsymbol{v}_i, \boldsymbol{p}^{(t)}) & : (\boldsymbol{v}_i \text{ with an 'anchor' label}) \end{cases} \tag{17}$$
where $c^{(t)}$ is the vertex label of $\boldsymbol{v}_c^{(t)}$. The learning rate $\epsilon(t)$, the neighborhood function $\lambda(i|c)$, and the translation vector $\Delta\boldsymbol{v}$ are obtained from Eqs.(9), (10), and (12).
6. Compute the signed distances between the moved vertices and their closest control points. If the distance is less than the threshold τ, the vertex is assigned to an 'anchor' label.
7. If no vertices are moved, or $t \geq T$, go to step 8. Otherwise, $t \leftarrow t + 1$ and go to step 3. Here, T is the maximum time for selecting winner vertices.
8. If the SDM includes foldovers, update the parameters according to Eq.(14), and go to step 2. Otherwise, go to step 9.
9. For each vertex $\boldsymbol{v}_i$:
   (a) By regarding the control point $\boldsymbol{p}$ included in $\boldsymbol{\Omega}_m$ as the corresponding point of $\boldsymbol{v}_i$, choose potential vectors $\tilde{\boldsymbol{v}}_u$ from $\tilde{\boldsymbol{v}}_u = \boldsymbol{v}_i + w(\boldsymbol{p} - \boldsymbol{v}_i)$ for updating the position of $\boldsymbol{v}_i$. The parameter w is a variable that gives the rate of moving $\boldsymbol{v}_i$ to $\boldsymbol{p}$.
   (b) Find the vector $\boldsymbol{v}_i^*$ satisfying
   $$\boldsymbol{v}_i^* = \arg \min_{\tilde{\boldsymbol{v}}_u} E(\tilde{\boldsymbol{v}}_u), \tag{18}$$
   where $E(\boldsymbol{v}_i)$ is the energy function in Eq.(15). In addition, update $\boldsymbol{v}_i$ from $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i^*$.
10. Compute the distance between the SDM and the target surface using Eq.(3). If the distance is less than a given threshold θ_e, the process is terminated. Otherwise, go to step 8.

Generally, in the SOM algorithm, the maximum time T for selecting winner vertices is set to a large value to achieve good accuracy for the SOM. In addition, the parameter T acts as a criterion for adopting the energy-based deformation process. Therefore, a large value for the parameter T leads to a deformation process that requires much computational time, even though the SDM is hardly deformed during the process. To avoid this situation, we use another criterion: when no vertices are moved, the algorithm is terminated regardless of the learning time t.

3.4 Modification of the SDM deformation algorithm

With the above algorithm, object mesh models used as the SDM can automatically be fitted to the given target surfaces. In addition, users can more easily control the deformation of the SDM more easily than the general deformable models. This characteristic provides the opportunity to obtain more preferable models depending on the applications using the models. Given below are some of the ways of modifying the algorithm for the SDM deformation.

3.4.1 Selection of a control point

After a winner vertex has been determined from a given control point, both the winner vertex and its neighbors move towards the control point. If some control points are chosen more often than others, more SDM vertices tend to move toward these control points. This means that the choice of the control points influences the density of vertices of the SDM on the target surface. Since the control points are randomly chosen, the choice can be controlled by changing the selection probability of the control points. For example, if all control points are chosen with the same probability, the vertices of the resulting SDM are located on the target surface according to the distribution of the control points. As a further example, if the selection probability of the control points from a particular region of the target surface is higher than that of other regions, the vertices of the SDM tend to move toward that particular region.

Let us consider the reconstruction of an object with a complex shape. Then, the reconstruction accuracy of the resulting model can be improved by moving more vertices of the SDM towards the area with the complex shape. One way of achieving this movement is to determine the selection probability of the control points based on the curvatures at the points. In practice, the selection probability P_u of control point $\boldsymbol{p}_u$ is computed as

$$P_u = \frac{C_u}{Z} + \beta \tag{19}$$

where C_u is the curvature at control point $\boldsymbol{p}$, and Z is a normalization factor such that $\sum P_u = 1$. The positive parameter β ($\beta > 0$) is used to set the selection possibility of all vertices with a non-zero value. Using the probabilities, control points are chosen by a roulette wheel selection which is one of the selection methods in generic algorithm. In a roulette wheel selection, control points with high selection probabilities are more likely to be selected, but there is also a chance of selecting control points with small probabilities.

3.4.2 Determination of a winner vertex

The SDM framework can move an arbitrary vertex towards a special control point as described below. This movement is achieved by introducing the constraint that the vertex is always selected as the winner of the special point. To begin with, users specify feature control points $\tilde{\boldsymbol{p}}$ and their corresponding vertices $\tilde{\boldsymbol{v}}(\tilde{\boldsymbol{p}})$. In the SDM deformation algorithm, if the t-th selected control point $\boldsymbol{p}^{(t)}$ is one of the feature points, its corresponding vertex is automatically

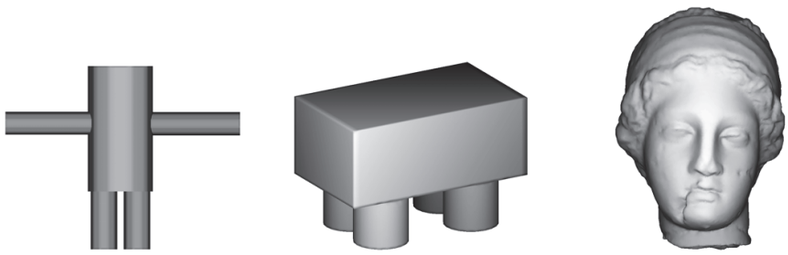
SDM ↓ target surface		pig ↓ sphere	rabbit ↓ cylindrical surface	armadillo ↓ Cylinderman	lion ↓ Animal	sphere ↓ Igea
SDM	vertices	3,516	67,037	16,980	9,154	10,242
	patches	7,028	134,070	33,947	18,304	20,480
target surface	points	2,526	79,602	11,305	8,602	33,587
(η_0, η_1)		(11, 5)	(10, 0)	(5, 7)	(11, 6)	(10, 0)
processing time [sec.]		61	286	1,055	644	231

Table 1. SDMs and target surfaces used in our experiments, and computational time for deforming the SDMs. The results except for the projection of the sphere to Igea are adapted from the article (Morooka et al., 2007). Copyright(c)2007 IEICE.

determined as the winner vertex. The determination is formulated by rewriting Eq.(16) as

$$\boldsymbol{v}_c^{(t)} = \begin{cases} \tilde{\boldsymbol{v}}(\boldsymbol{p}^{(t)}) & : (\boldsymbol{p}^{(t)} \in \mathcal{F}) \\ \arg\min_{\{i\} \in \mathcal{K}_v} SD(\boldsymbol{v}_i, \boldsymbol{p}^{(t)}) & : (\text{otherwise}), \end{cases} \tag{20}$$

where $\mathcal{F}$ is the set of the pairs $(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{v}}(\tilde{\boldsymbol{p}}))$ of the feature points $\tilde{\boldsymbol{p}}$ and their corresponding vertices $\tilde{\boldsymbol{v}}(\tilde{\boldsymbol{p}})$. Under this constraint, the SDM is deformed in such a way that the vertex moves constantly towards the control point.



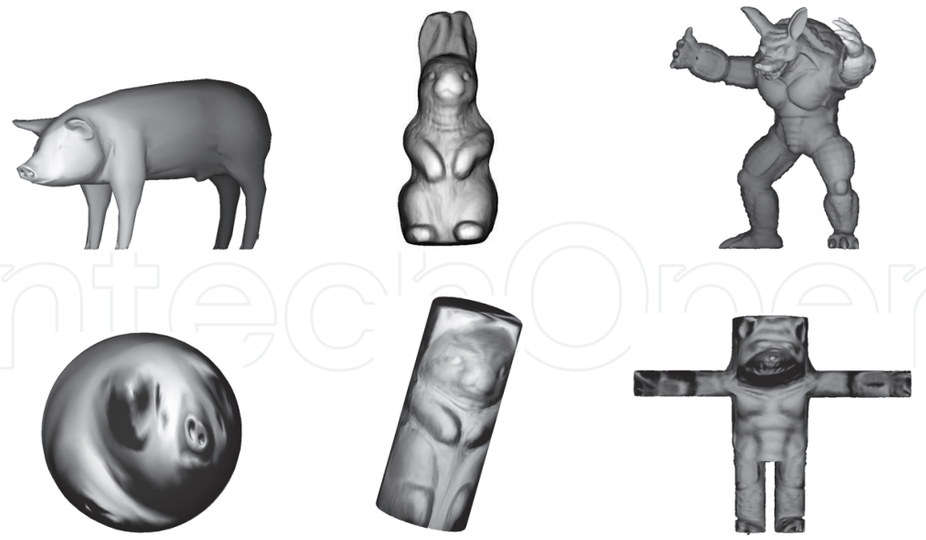
(a)Cylinderman (b)Animal (c)Igea

Fig. 4. Examples of target surfaces. Fig.4(a) and (b) are adapted from the article (Morooka et al., 2007). Copyright(c)2007 IEICE.

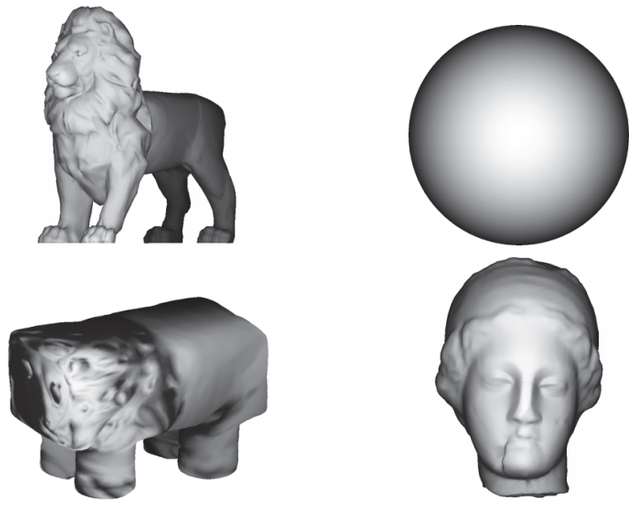
4. Experimental results

To verify the applicability of our proposed method, we conducted several experiments that involved applying the SDM to various 3D object models. The well-known "rabbit", "Igea", and "armadillo" were included as models in our experiments. These models were downloaded from Cyberware and Stanford University websites. We set the parameters in the SDM deformation as follows:

$$\begin{aligned} \epsilon_s &= 0.5, & \epsilon_d &= 0.05, & \sigma_s &= 3.0, \\ \sigma_d &= 0.1, & w &= 0.01, & \theta_e &= 0.2. \end{aligned} \tag{21}$$



(a) pig - spherical surface (b) rabbit - cylindrical surface (c) armadillo - Cylinderman



(d) lion - Animal (e) spherical surface - Igea

Fig. 5. Projection of SDMs onto various kinds of target surfaces; top : initial SDMs; bottom : SDMs projected onto target surfaces. Fig.5(a)-(d) are adapted from the article (Morooka et al., 2007). Copyright(c)2007 IEICE.

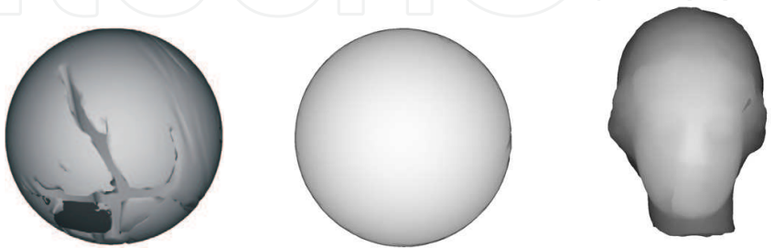
Given the target surface with N_v vertices, the maximum number T of selecting the control points is set to $\eta_0 N_v$. We denote as a parameter η_1 the repeat count of restarting the learning process. Finally, $(\eta_0 + \eta_1)N_v$ control points are selected in the SDM deformation. These parameters used in our experiments are shown in Table 1.

4.1 Fitting of SDMs to target surfaces

In the first experiment we project a mesh model onto another kind of target surface. The target surfaces used in our experiments are a spherical surface, a cylindrical surface, a roughly quadrupedal surface, and a simple humanoid surface. The latter two models, shown in Fig.

(a) (b) (c)

Fig. 6. Facial part of armadillo projected onto Cylinderman. Adapted from the article (Morooka et al., 2007). Copyright(c)2007 IEICE.



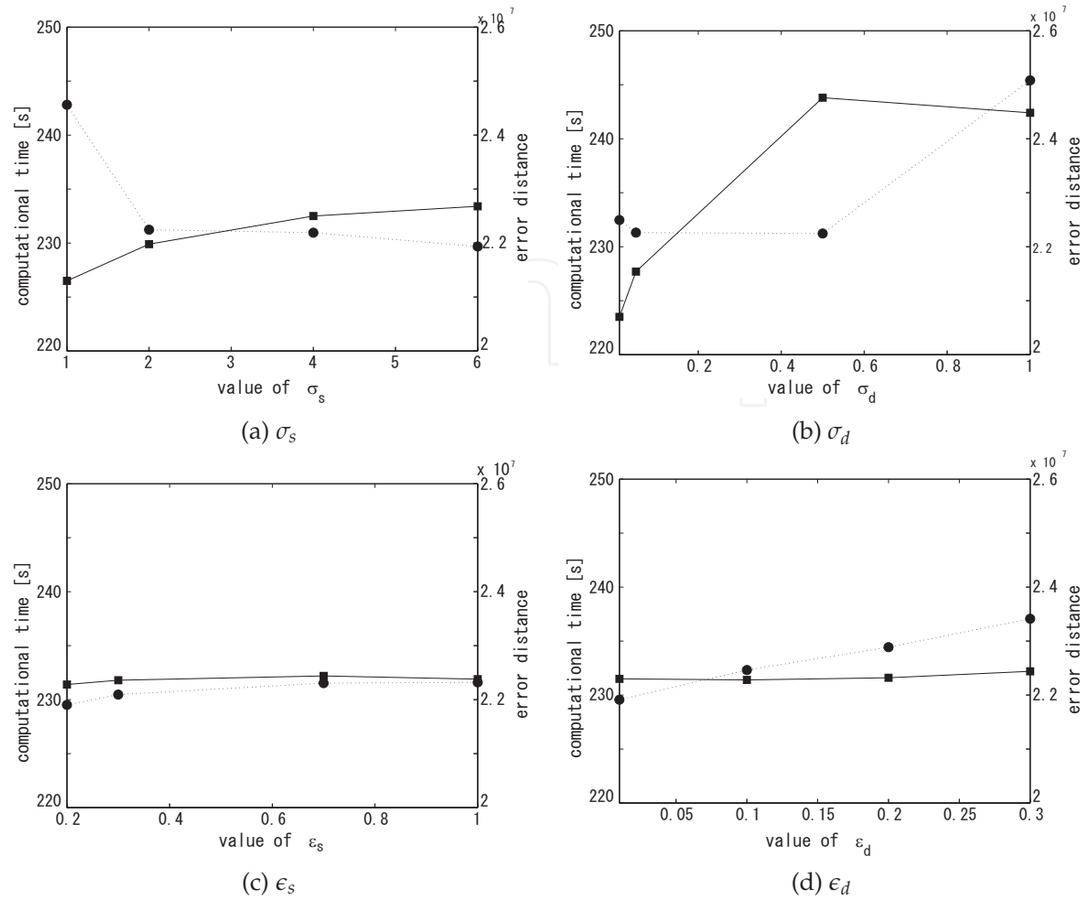
(a) (b) (c)

Fig. 7. The models obtained using the original SOM algorithm.

4, are called “Animal” and “Cylinderman”. Since all the target surfaces are represented by parametric functions, the control points needed to represent each surface are calculated from the functions. Table 1 shows the number of vertices and faces of the initial SDMs, and the number of control points together with the computational time for each projection.

The top row in Fig. 5 shows the initial SDMs, while the bottom row displays the projected models. In each of these figures, the normal of the initial SDM is mapped onto the final SDM and its shape approximates the target surface. Amongst these experiments, the projection of the “armadillo” model onto the Cylinderman proved to be the most time-consuming because of the complex shapes. Fig. 6 shows the facial part of the projected armadillo in shading and wireframe representations. Fig. 6(a) displays an enlarged view of the model shown in Fig. 5(c), while the shading model shown in Fig. 6(b) is obtained using normals of the final SDM. The SDM can be projected onto the target surface smoothly without foldovers. It is clear from these results that our method can project mesh models onto various kinds of mapping surfaces in a unified framework compared with previous works that required mapping functions.

Next, we compare our proposed method to that which applies the original SOM algorithm to an SDM deformation. We refer to the latter method as the comparison method. The pig model used in the first experiment is projected onto a spherical surface using the comparison method. Fig. 7(a) is the shading model obtained from the comparison method. The model is projected onto a limited area of the target surface. Some unnatural edges are seen on the model surface. By contrast, Fig. 7(b) is the shading model given in Fig. 5(a). Since the SDM deformation introduces certain processes to avoid the collapse problem, the SDM can be projected onto the target surface while not containing any foldover areas. Next, we fit a spherical surface model as an SDM to the female model ‘Igea’ as shown in Fig. 4(c). The purpose of the SOM algorithm is to represent the distribution of the control points by the SOM. Therefore, in the comparison



(a) σ_s (b) σ_d

(c) ϵ_s (d) ϵ_d

Fig. 8. Comparison of the computational time and the error distance in Eq.(3) in the SDM deformation by changing the parameters: (a)σ_s, (b)σ_d, (c)ϵ_s and (d)ϵ_d.

method, the final SDM as shown in Fig. 7(c) is blurred and incomplete. Fig. 5(e) displays the model obtained using our proposed method. According to these figures, our method can approximate the original shape.

With reference to the parameters in the SDM deformation, we verified the sensitivity of our method to these parameters. This verification uses the example mapping from a spherical surface as the initial SDM to the Igea as the target surface in Fig. 5 (e). In this experiment, the four main parameters σ_s, σ_d, ϵ_s and ϵ_d are used, and their values in Eq.(21) are regarded as the original ones. Only one parameter is set to a value selected from the following sets:

$$\begin{aligned} \sigma_s &= \{1.0, 2.0, 4.0, 6.0\}, & \sigma_d &= \{0.01, 0.05, 0.5, 1.0\}, \\ \epsilon_s &= \{0.2, 0.3, 0.7, 1.0\}, & \epsilon_d &= \{0.01, 0.1, 0.2, 0.3\}, \end{aligned} \tag{22}$$

while the other parameters use the original values. The SDM deformation is performed by using the various parameter sets. Fig. 8 shows the computational time and the error distance between the final SDM and the target surface in Eq.(3) when changing each parameter. In this

figure, the computational time and the error are represented by a straight line and a dotted one. The computational time and the error using the original values of the parameters are 231.4 [sec] and 2.2×10^{-7}. Noting these results, the accuracy of all the final SDMs is good regardless of the settings of the parameters. On the other hand, the computational time and the error tend to increase in the SDM deformation using the parameters so that the ratio σ_d/σ_s or ϵ_d/ϵ_s is large. The reason for this is that the parameters leads to the retention of a large adaptation rate or the large learning rate during the deformation process, and to a slow convergence of the SDM deformation.

In our experiments, we determined the parameters through some preliminary experimental work. The SDM deformation is based on the SOM algorithm, which contains the parameters such as the learning rate in Eq.(9) and the neighborhood radius in Eq.(11). However, there is no theoretical framework for the optimal selection of the parameters included in the SOM algorithm. For optimal selection, one way is to incorporate the Bayesian framework into the SOM algorithm (Bishop et al., 1998; Utsugi, 1996; 1997). This incorporation enables us to regard the SOM algorithm as an estimating algorithm for the Gaussian mixture model. Given the input data, self-organizing mapping is performed by estimating a probability distribution, generating the data by using the expectation-maximization algorithm. The optimal hyperparameters in the model can be estimated from the data. Our future work will include the extension of the SDM using the Bayesian framework.

As shown in Table 1, the deformations of the SDMs with complex shapes require more computational time than other SDM deformations. Since the model obtained by the SDM tends to include a foldover, the SDM deformation is restarted several times to remove the foldover from the SDM. Our future work includes the development of a method to reduce the computational time of the SDM deformation.

4.2 Control of deforming SDM

In Section 3.4, we noted some of the advantages of the SDM deformation. This experiment was conducted to verify the possibility of controlling the SDM deformation. To begin with, we acquired facial range data from 29 males using a range finder. The range data of each face was composed of about 85,000 points, and the points were used directly as control points. A model of each face was obtained by fitting a spherical SDM to the facial range data. Then, we set the single constraint that certain specific feature vertices of the SDM always moved towards their corresponding control points one by one. In practice, we manually chose 12 vertices in the SDM. These are illustrated in Fig. 9(a) as black points. We also selected 12 control points by hand from the facial range data as shown in Fig. 9(a). If one of these control points was chosen in step 3 of the SDM deformation algorithm, its corresponding vertex was automatically determined as the winner vertex.

Fig. 9(b)-(d) shows the process of deforming the sphere under the constraint, while Fig. 9(d) shows the final model. Fig. 9(e) shows another facial model generated by deforming the SDM without the constraint. Observation of these figures show that the final model recovers the shape of the face while the twelve feature vertices in the final model are located at their target points. The 29 face models are obtained under the above constraint. Since the models obtained contain the same number of vertices and the same topological information, we can easily find

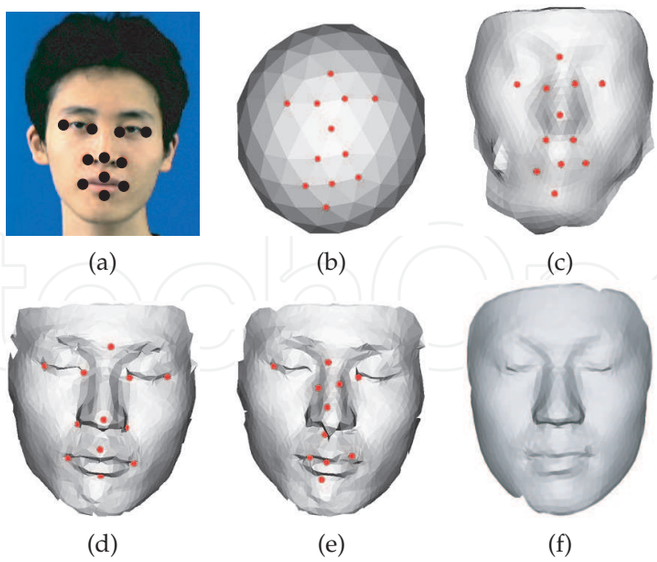


Fig. 9. Control of deforming SDM: (a) the target points of a male face on a photographic image; (b-d) the process of deforming the SDM under the constraint that some vertices of the SDM correspond to control points on the target surface; (e) the facial surface recovered from spherical mesh model without the constraint; (f) the average model of 29 faces. Adapted from the article (Morooka et al., 2007). Copyright(c)2007 IEICE.

Fig. 10. Control of deforming SDM by changing the selection probability of control points.

a structural correspondence between any two facial models. This correspondence is useful to computer vision and computer graphics applications. As an example of an application, we generate the average model of male faces by computing the average position of each vertex with the same label. Fig. 9(f) shows the average facial mesh model.

Also the SDM deformation can be controlled by changing the selection possibility of the control points. In this experiment, a spherical surface and a female face as shown in Fig. 10(a) are used as the initial SDM and target surface, respectively. The target surface is resampled so that the control points are distributed uniformly. From Eq.(19), we determine the selection possibility of the control point. The determination uses the average curvature at the control point $\boldsymbol{p}$ as the curvature $C(\boldsymbol{p})$ in Eq.(19). The average curvature is computed by the method in (Hameiri et al., 2002). Fig. 10(b) shows the visualization of the high average curvature of the face. In particular, the eyes, nose, and mouth areas contain points with high curvatures.

After the SDM deformation, the final SDM is obtained as shown in Fig. 10(c). The distribution of the vertices is represented by the average distance A_i between each vertex $\boldsymbol{v}_i$ and its 1-ring neighbors. When many vertices are intensively collected in certain regions, the average distance between the vertices is small. On the other hand, the average distance is large when regions contain a small number of vertices. Using the average distance, the red color R_i of the vertex $\boldsymbol{v}_i$ is assigned by

$$R_i = \begin{cases} (A_i - A_{min})/(A_{max} - A_{min}) & : (A_i < \text{threshold}) \\ 0 & : (\text{otherwise}), \end{cases} \tag{23}$$

where A_{min} and A_{max} are, respectively, the minimum and maximum average distances between the vertices whose A_i is less than a given threshold. Fig. 10(c) shows the final SDM, while Fig. 10(d) shows the visualization of the distribution of the vertices in the final SDM. Comparing Figs.10(b)-(d), the SDM deformation can recover the target surface while locating many vertices in the regions with high curvatures by controlling the selection possibilities of the vertices.

5. Conclusion

This paper proposed a method for fitting mesh models to a given surface using the Self-organizing Deformable Model (SDM). The basic idea behind the SDM is the combination of competitive learning and energy minimization. Although the SDM and the target surface must have the same topological type, there are few other constraints when using the SDM. Our SDM can be applied to various target surfaces, whereas traditional methods using mapping functions only deal with a specific target surface. Furthermore, compared with deformable models based on energy minimization, our proposed method enables us to control the SDM deformation easily by changing the ways in which control points are chosen and a winner vertex is determined. The experimental results show that our SDM is a powerful tool for use in various fields in computer vision and computer graphics. For example, the SDM enables us to establish the correspondence between two mesh models by mapping them onto a common target surface. We are currently developing SDM-based applications.

6. Acknowledgment

This work was supported by a Grant-in-Aid for Scientific Research(C) (No. 23500244) of The Ministry of Education, Culture, Sports, Science, and Technology, Japan.

7. References

Allen, B.; Curless, B.; & Popović, Z. (2003). The space of human body shapes: reconstruction and parameterization from range scans, *Proc. SIGGRAPH '03*, pp.587–594, 2003.

Barhak, J.; & Fischer, A. (2001). Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques, *IEEE Transactions on Visualization and Computer Graphics*, vol.7, no.1, pp.1–16, 2001.

Bishop, C.M.; Svensén, M.; & Williams, C.K.I. (1998). GTM : the generative topographic mapping, *Neural Computation*, vol.10, no.1, pp.215–235, 1998.

Blanz, V.; Mehl, A.; Vetter, T.; & Seidel, H.P. (2004). A statistical method for robust 3d surface reconstruction from sparse data, *Proc. of the 3D Data Processing, Visualization, and Transmission*, pp.293–300, 2004.

Cignoni, P.; Rocchini, C.; & Scopigno, R. (1998). Metro: Measuring error on simplified surfaces, *Computer Graphics Forum*, vol.17, no.2, pp.167–174, 1998.

Duan, Y.; & Qin, H. (2004). A subdivision-based deformable model for surface reconstruction of unknown topology, *Graphcal Models*, 66(4), 181–202 (2004)

Duda, R.O.; Hart, P.E.; & Stork, D.G. (2000). *Pattern Classification (2nd Edition)*, Wiley-Interscience.

Eck, M.; DeRose, T.; Duchamp, T.; Hoppe, H.; Lounsbery, M.; & Stuetzle, W. (1995). Multiresolution analysis of arbitrary meshes, *Proc. SIGGRAPH '95*, vol. 29, pp. 173–182, 1995.

Floater, M.; & Hormann, K. (2003). Recent advances in surface parameterization, *Multiresolution in Geometric Modeling 2003*, pp. 259–284, 2003.

Fritzke, B. (1997). Some competitive learning methods, 1997.

Gibson, S.; & Mirtich, B. (1997). A survey of deformable modeling in computer graphics, *Technical Report TR97-19*, Mitsubishi Electric Research Lab. (1997)

Hameiri, E.; & Shimshoni, I. (2002). Estimating the Principal Curvatures and the Darboux Frame from Real 3D Range Data, *Proc. 3DPVT 2002*, pp. 258–267, 2002.

Ivrissimtzis, I.; Lee, Y.; Lee, S.; Jeong, W.K.; & Seidel, H.P. (2004). Neural mesh ensembles, *Proc. Int. Symp. on 3DPVT*, pp. 308–315, 2004.

Ivrissimtzis, I.P.; Jeong, W.K.; & Seidel, H.P. (2003). Using growing cell structures for surface reconstruction, *Shape Modeling International 2003*, pp. 78–88, 2003.

Ju, T.; Schaefer, S. & Warren, J. (2005). Mean value coordinates for closed triangular meshes, *ACM Trans. Graph.*, vol. 24, no. 3, pp. 561–566, 2005.

Kavan, L.; & Zara, J. (2005). Spherical blend skinning: a real-time deformation of articulated models, *Proc. the 2005 symposium on Interactive 3D graphics and games*, pp. 9–16, 2005.

Kohonen, T. (2001). *Self-Organizing Maps*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

Kraevoy, V.; & Sheffer, (2004). A. Cross-parameterization and compatible remeshing of 3D models, *Proc. SIGGRAPH '04*, pp. 861–869, 2004.

Kun, Y.; Zhou, K.; Xu, D.; Shi, X.; Bao, H.; Guo, B.; & Shum, H.Y. (2004). Mesh editing with Poisson-based gradient field manipulation, *Proc. SIGGRAPH '04*, pp. 644–651, 2004.

Kurazume, R.; Nakamura, K.; Okada, T.; Sato, Y.; Sugano, N.; Koyama, T.; Iwashita, Y.; & Hasegawa, T. (2007). 3D reconstruction of a femoral shape using a parametric model and two 2d fluoroscopic images, *Proc. ICRA'07*, pp. 3002–3008, 2007.

Lachaud, J.O.; & Montanvert, A. (1999). Deformable meshes with automated topology changes for coarse-to-fine 3d surface extraction, *Medical Image Analysis*, vol. 3, no. 2, pp. 187–207, 1999.

Lee, A.W.F.; Sweldens, W.; Schröder, P.; Cowsar, L.; & Dobkin, D. (1998). MAPS: Multiresolution adaptive parameterization of surfaces, *Proc. SIGGRAPH '98*, pp. 95–104, 1998.

Liou, C.Y.; & Kuo, Y.T. (2005). Conformal self-organizing map for a genus-zero manifold, *The Visual Computer*, vol. 21, no. 5, pp. 340–353 , 2005.

Lipman, Y.; Sorkine, O.; Alexa, M.; Cohen-Or, D.; Levin, D.; Rössl, C.; & Seidel, H.P. (2005). Laplacian framework for interactive mesh editing, *International Journal of Shape Modeling*, vol. 11, no. 1, pp. 43–62 , 2005.

McInerney, T.; & Terzopoulos, D. (2000). T-snakes: Topology adaptive snakes, *Medical Image Analysis*, vol. 4, no. 2, pp. 73–91, 2000.

Morooka, K.; & Nagahashi, H. (2005). Self-organizing Deformable Model : A New Method for Fitting Mesh Model to Given Object Surface, *Proc. of Int. Symp. Visual Computing*, pp. 151–158, 2005.

Morooka, K.; Matsui, S.; & Nagahashi, H. (2007). Self-organizing Deformable Model : A New Method for Projecting Mesh Model of 3D Object onto Target Surface, *IEICE Transactions on Information and Systems (Japanese Edition)*, vol. J90-D, no. 3, pp.908–917, 2007.

Noh, J.Y.; & Neumann, U. (2001). Expression cloning, *Proc. SIGGRAPH '01*, pp. 277–288, 2001.

Schreiner, J.; Asirvatham, A.; Praun, E.; & Hoppe, H. (2004). Inter-surface mapping, *Proc. SIGGRAPH '04*, pp. 870–877, 2004.

Sederberg, T. W.; & Parry, S. R. (1986). Free-form deformation of solid geometric models, *Proc. SIGGRAPH '86*, pp. 151–160, 1986.

Sumner, R.W.; & Popovic, J. (2004). Deformation transfer for triangle meshes. *Proc. SIGGRAPH '04*, pp. 399–405, 2004.

Surazhsky, V.; Surazhsky, T.; Kirsanov, D.; Gortler, S.; & Hoppe, H. (2005). Fast exact and approximate geodesics on meshes, *Proc. SIGGRAPH '05*, pp. 553–560, 2005.

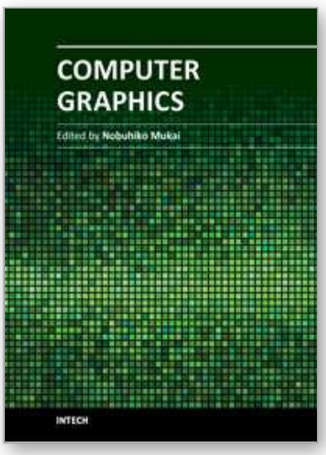
Tarini, M.; Hormann, K.; Cignoni, P.; & Montani, C. (2004). Polycube-maps, *Proc. SIGGRAPH '04*, pp. 853–860, 2004.

Utsugi, U. (1996). Topology selection for self-organizing maps, *Network: Computation in Neural Systems*, vol. 7, no. 4, pp. 727–740, 1996.

Utsugi, U. (1997). Hyperparameter selection for self-organizing maps, *Neural Computation*, vol. 9, no. 3, pp. 623–635, 1997.

Yu, Y. (1999). Surface reconstruction from unorganized points using self-organizing neural networks, *Proc. IEEE Visualization '99*, pp. 61–64, 1999.

Zayer, R.; Rössl, C.; Karni, Z.; & Seidel, H.P. (2005). Harmonic guidance for surface deformation, *Comput. Graph. Forum*, pp. 601–609, 2005.



Computer Graphics

Edited by Prof. Nobuhiko Mukai

ISBN 978-953-51-0455-1
Hard cover, 256 pages
Publisher InTech
Published online 30, March, 2012
Published in print edition March, 2012

Computer graphics is now used in various fields; for industrial, educational, medical and entertainment purposes. The aim of computer graphics is to visualize real objects and imaginary or other abstract items. In order to visualize various things, many technologies are necessary and they are mainly divided into two types in computer graphics: modeling and rendering technologies. This book covers the most advanced technologies for both types. It also includes some visualization techniques and applications for motion blur, virtual agents and historical textiles. This book provides useful insights for researchers in computer graphics.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Ken'Ichi Morooka and Hiroshi Nagahashi (2012). Self-organizing Deformable Model : a Method for Projecting a 3D Object Mesh Model onto a Target Surface, Computer Graphics, Prof. Nobuhiko Mukai (Ed.), ISBN: 978-953-51-0455-1, InTech, Available from: http://www.intechopen.com/books/computer-graphics/self-organizing-deformable-model-a-method-for-projecting-a-3d-object-mesh-model-onto-a-target-surfac

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen