Feature-preserving Deformation for Assembly Models

Hiroshi Masuda

The University of Tokyo, masuda@nakl.t.u-tokyo.ac.jp

ABSTRACT

This paper proposes a framework for interactively deforming assembly models with multiple disconnected meshes. When the user selects a mesh model and deforms it, other mesh models are simultaneously and consistently deformed while the shapes of the form-features are preserved. The method combines surface-based and volume-based deformation. Each model is encoded using differential properties for the surface-based deformation and mean value coordinates for the volume-based deformation. Since the deformation methods are interrelated through linear systems, the deformation of one model can be interactively propagated to the other models. The implemented system can achieve a real-time response for the deformation of assembly models.

Keywords: Interactive deformation, feature-preserving deformation, discrete Laplacian, mean curvature, mean value coordinates.

1. INTRODUCTION

Surface-based mesh deformation techniques have been intensively studied [1–5]. In such techniques, the shape of a mesh model is encoded using differential equations, and it is deformed by modifying the boundary conditions of the differential equations. When differential equations are approximated as a linear system [3–5], they can be interactively solved using state-of-the-art linear solvers [6]. In such interactive deformation, the user first selects the fixed region, which remains unchanged, and the handle region, which is used as the manipulation handle, and interactively deforms the shape by dragging the handle over the screen.

Surface-based deformation techniques are suitable for specifying various constraints at vertices, edges, and faces on a mesh model. However, since surface-based deformation encodes the detail of the shape using the topological connectivity, it cannot propagate deformation to disconnected models. When this technique is applied to an assembly model with disconnected components, tedious manual work is needed to consistently deform the multiple mesh models. On the other hand, volume-based deformation techniques, such as free-form deformation (FFD) [7–9], change geometric shapes by deforming the space in which the object lies. Volume-based deformation can simultaneously deform several disconnected models inside the volumetric space. However, it is difficult to manage the constraints at the vertices, edges and faces on mesh models, because this technique does not directly deform the mesh models. Therefore, volume-based techniques often modify the shapes of the product in unintended ways, for example, circular holes change into ellipses.

Our motivation for studying interactive deformation stems from the requirements for the deformation of automobile sheet metal panels. Sheet metal panels typically consist of free-form surfaces and form-features [10,11]. In sheet metal panels, some curves and surfaces are often required to keep their original shapes, such as circles and cylinders. Such partial regions are called *form-features* in product design. In addition, sheet metal panels are typically assembled into a modular component by welding or bolting. When mesh models are used for CAE applications, individual mesh models are combined in a single file with the attributes of the contact types, such as welded or bonded. When mesh models are deformed to evaluate various shapes, the disconnected mesh models should be simultaneously and consistently deformed. Although volume-based deformation can deform disconnected models, it cannot precisely preserve engineering constraints, such as the shapes of the form-features. Surface-based deformation is suitable for preserving engineering constraints; however, it cannot be applied to disconnected mesh models.

In our previous work [12], we extended surface-based deformation techniques to precisely preserve the shapes of the form-features during deformation. We specified the differential properties as soft constraints, which are approximately satisfied in the least squares sense, and the user-defined constraints, including the positions of the fixed regions and handle regions, as hard constraints, which must be precisely satisfied. We solved those constraints using the Lagrange multiplier method. Our method could successfully deform sheet metal panels while preserving the form-features.

In this paper, we propose a new interactive deformation framework that can be applied to assembly models. Our method propagates the deformation of one model to other models using a combination of surface-based deformation and volume-based deformation. Our volume-based deformation is based on mean value coordinates. The mean value coordinates were originally proposed by Floater [13] for smoothly interpolating the inside of a non-convex polygon. This idea was extended to 3D space inside a closed triangular mesh by Ju et al. [14] and Floater et al. [15]. This technique represents the position $\mathbf{p}$ using a linear combination of the control points $\{\mathbf{q}_i\}$ and deforms mesh models by manipulating the control points.

Huang et al. [16] applied mean value coordinates to surface-based deformation; they represented all the vertex positions of the mesh models as the weighted sums of the control points, and substituted them into linear differential equations for surface-based deformation. Since the vertex positions of the disconnected models can be represented using the common control points, it is possible to deform the disconnected models simultaneously. Their method is very smart, but the degree of freedom is restricted to the number of vertices on a closed mesh. If the number of vertices is increased, the degree of freedom is also increased; however, the large number of vertices significantly deteriorates the performance, because linear systems for mean value coordinates are dense matrices, which are calculated at a cost of O(n^3). Even if the matrices are factorized to improve the performance, the calculation cost is still O(n^2). Therefore, it is actually impossible for their method to precisely and interactively satisfy the various engineering constraints. Zhou et al. [17] proposed a method to encode a volumetric region using volumetric Laplacian equations, but it is not easy to apply their method to assembly models. To the present author's knowledge, no methods have been proposed for interactively deforming assembly models and precisely preserving the engineering constraints.

We use mean value coordinates to make the connection for the disconnected mesh models. In our deformation framework, the control points of a closed mesh are determined using a mesh model that is deformed using surface-based deformation, and the control points are used to deform the boundary conditions on other mesh models.

The main contribution in this paper is to propose: first, a novel deformation framework for simultaneously deforming an assembly model with disconnected components while preserving the form-features; secondly, a stable and efficient computation method for the control points of volume-based deformation; and thirdly, the introduction of local control meshes for propagating deformation to the contact regions in an assembly model.

In the following section, we describe the surface-based deformation and volume-based deformation, which are used in our framework. In Section 3, we describe our mesh deformation framework. In Section 4, we explain our deformation method for assembly models and show experimental results. We conclude the paper in Section 5.

2. INTERACTIVE MESH DEFORMATION

Our deformation framework consists of a combination of surface-based and volume-based deformation. In this section, we explain both deformation techniques.

2.1 Surface-based Feature-preserving Deformation

Masuda et al. [12] proposed an interactive mesh deformation technique that precisely preserves the shapes of the form-features. We extend this method so that disconnected models can be simultaneously deformed. First we explain our feature-preserving deformation described in [12]. We simply call this method *feature-preserving deformation*.

Fig. 1 shows examples of feature-preserving mesh deformation. Surface-based deformation often deforms the shapes of the form-features in undesirable ways, as shown in Fig. 1(b). Feature-preserving deformation preserves the shapes of the form-features during interactive deformation, as shown in Fig. 1(c, d).

Let the mesh M be a pair $\{K, \mathbf{P}\}$, where K is a simplicial complex, which consists of vertices i, edges (i,j), and faces (i,j,k); $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ is the vertex positions of the mesh in $\mathbb{R}^3$. The adjacent vertices of vertex i are denoted by $N(i) = \{j \mid (i,j) \in K\}$. The original position of $\mathbf{p}_i$ is referred to as $\hat{\mathbf{p}}_i$.

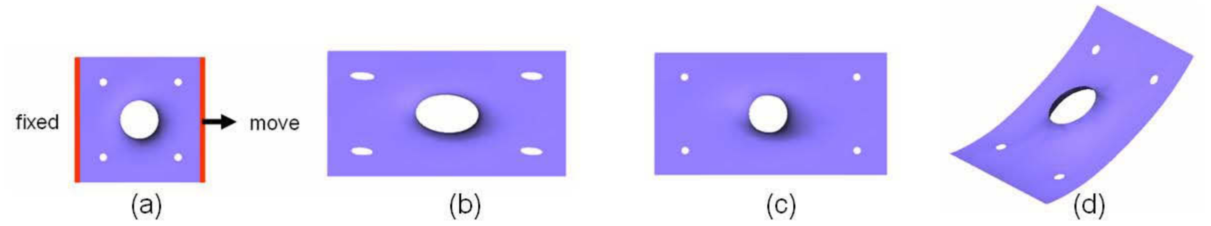


Fig. 1: Feature-preserving deformation. (a) Original shape; (b) deformed shape with no feature constraints; (c) the shapes of circles are preserved; (d) five circles are rotated while preserving their shapes.

The normal vector and mean curvature at vertex i are referred to as κ_i and $\mathbf{n}_i$, respectively. According to [18–20], the discrete mean curvature normal $\kappa_i \mathbf{n}_i$ can be represented as:

$$\kappa_i \mathbf{n}_i = \mathbf{L}(\mathbf{p}_i) \equiv \frac{1}{4A_i} \sum_{j \in N(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{p}_i - \mathbf{p}_j), \tag{2.1}$$

where α_{ij} and β_{ij} are the two angles opposite to the edge in the two triangles that share edge (i, j). When the mean curvature normals of the original mesh are denoted by $\{\delta_1, \delta_2, \ldots, \delta_n\}$, the error metrics of the mean curvatures are defined as:

$$\sum_{i=1}^{n} (\mathbf{L}(\mathbf{p}_i) - R_i \delta_i)^2, \tag{2.2}$$

where R_i is the rotation matrix for the normal vector $\mathbf{n}_i$. R_i is calculated at each vertex before the vertex positions are calculated. The minimization of Eqn. (2.2) produces the thin-plate energy-minimization surface. See [4,19] for the detail discussion of energy minimization.

When the user specifies the coordinates of the points on a mesh, the following *positional constraints* are added as boundary conditions:

$$f_l(\mathbf{P}) \equiv s_l \mathbf{p}_i + t_l \mathbf{p}_j + u_l \mathbf{p}_k = \mathbf{u}_l \qquad (1 \le l \le n_p,\ s_l + t_l + u_l = 1), \tag{2.3}$$

where $\mathbf{u}_l$ is any point specified by the user; $\mathbf{p}_i$, $\mathbf{p}_j$ and $\mathbf{p}_k$ are the vertex positions of a triangle that includes the specified point on the mesh; n_p is the number of positional constraints.

We define a form-feature as a partial shape that has an engineering meaning and represent it as a subset of $(K, \mathbf{P})$. The following constraints then maintain the shape of the form-feature k.

$$\mathbf{p}_i - \mathbf{p}_j = R_i(\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j) \qquad (\ (i,j) \in F_k,\ 1 \le k \le n_f\ ), \tag{2.4}$$

where R_i is a rotation matrix at vertex i; F_k is a set of edges in the form-feature k; n_f is the number of form-features.

The rotation matrix R_i can be determined using rotation axis $\mathbf{v}_i$ and angle θ_i. It is well known that the rotation can be represented using a unit quaternion by regarding $\mathbf{v}_i$ as three distinct imaginary numbers:

$$Q_i = \cos\frac{\theta}{2} + \mathbf{v}_i \sin\frac{\theta}{2} = \exp\frac{\theta}{2}\mathbf{v}_i. \tag{2.5}$$

The logarithm of the unit quaternion is defined as the inverse of the exponential:

$$\mathbf{r}_i = \ln Q_i = \frac{\theta}{2}\mathbf{v}_i. \tag{2.6}$$

We assign the quaternion logarithm $\mathbf{r}_i$ to vertex i and then introduce the following equations for smoothly interpolating quaternion logarithms:

$$\mathbf{L}(\mathbf{r}_i) = 0 \qquad (1 \le i \le n). \tag{2.7}$$

These equations produce the harmonic interpolation of quaternion logarithms over the mesh model. For the positional constraints and feature constraints, the following two types of constraints are added to rotations:

$$\begin{cases} f_j(\mathbf{r}) = \mathbf{c}_j & (\ 1 \le j \le n_p\ ) \\ \mathbf{r}_i - \mathbf{r}_j = 0 & (\ (i,j) \in F_k,\ \ 1 \le k \le n_f\ ) \end{cases}, \tag{2.8}$$

where $\mathbf{c}_j$ is a user-defined rotation. The second equation is required to maintain the original shape of a form-feature.

The rotations and coordinates can be calculated by solving the following two optimization problems:

$$\min\left(\sum_{i=1}^{n} \left\|\mathbf{L}(\mathbf{r}_i)\right\|^2\right) \quad \text{subject to} \begin{cases} f_j(\mathbf{r}) = \mathbf{c}_j & (1 \le \mathrm{j} \le n_p) \\ \mathbf{r}_i - \mathbf{r}_j = 0 & ((i,j) \in F_k,\ \ 1 \le k \le n_f) \end{cases}, \tag{2.9}$$

$$\min\left(\sum_{i\in\mathbf{\Lambda}} \left\|\mathbf{L}(\mathbf{p}_i) - R_i\delta_i\right\|^2\right) \quad \text{subject to} \begin{cases} f_i(\mathbf{P}) = \mathbf{u}_i & (i \in \mathbf{\Lambda}_p) \\ \mathbf{p}_i - \mathbf{p}_j = R_i(\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j) & ((i,j) \in F_k,\ \ 1 \le k \le n_f) \end{cases}. \tag{2.10}$$

The solution of Eqn. (2.9) is used to calculate R_i in Eqn. (2.10), which is the rotation matrix that rotates vector δ_i around axis $\mathbf{r}_i / |\mathbf{r}_i|$ by angle $2|\mathbf{r}_i|$. Eqns. (2.9) and (2.10) can be solved using the Lagrange multiplier method and converted into linear systems with sparse matrices, which can be efficiently factorized using the SuperLU solver [21]. The rotations and positions can be solved interactively when the matrix has been factorized once.

2.2 Volume-based Deformation Based on Mean Value Coordinates

We use mean value coordinates (MVC) to propagate the deformation of one mesh model to other disconnected models. The volume-based deformation based on the MVC is termed the *MVC-based deformation*; a closed triangular mesh for MVC is termed a *control mesh*; the mesh models inside the control mesh are termed *target models*.

We represent point $\mathbf{p}_i \in \mathbb{R}^3$ as the following weighted sum of the vertex positions $\{\mathbf{q}_j\}$ of the control mesh.

$$\mathbf{p}_i = \sum_{j=1}^{m} w_{ij}\mathbf{q}_j \qquad (\sum_{j=1}^{m} w_{ij} = 1), \tag{2.11}$$

where $\{w_{ij}\}$ are the MVC. Once the MVC have been calculated, the vertex positions inside the control mesh can be deformed by manipulating the control mesh [14,15]. For calculating the MVC, suppose a unit sphere $U(\mathbf{p}_i)$ is centered at $\mathbf{p}_i$. Any point on control mesh S can be parameterized on the unit sphere by projecting the point on $U(\mathbf{p}_i)$. We represent the projection of S on $U(\mathbf{p}_i)$ as $\tilde{S}$. Then the following equation is generally satisfied for the control mesh S, because the integral of the unit normal vectors on a sphere is $\mathbf{0}$.

$$\int_{\tilde{S}} \frac{\mathbf{q} - \mathbf{p}_i}{sign(\mathbf{q}) \cdot |\mathbf{q} - \mathbf{p}_i|} d\tilde{S} = \mathbf{0} \quad , \tag{2.12}$$

where $sign(\mathbf{q})$ is the sign of the inner product $(\mathbf{n}, \mathbf{q} - \mathbf{p}_i)$; $\mathbf{n}$ is the normal vector of the control mesh at $\mathbf{q}$. When surface S consists of a set of triangles $\{T_i\}$, this equation can be converted as follows:

$$\mathbf{p}_i = \int_{\tilde{S}} \frac{\mathbf{q}}{dist(\mathbf{q})} d\tilde{S} \bigg/ \int_{\tilde{S}} \frac{1}{dist(\mathbf{q})} d\tilde{S} = \left(\sum_j \int_{\tilde{T}_j} \frac{\mathbf{q}}{dist(\mathbf{q})} d\tilde{T}_i\right) \bigg/ \int_{\tilde{S}} \frac{1}{dist(\mathbf{q})} d\tilde{S} \quad , \tag{2.13}$$

where $dist(\mathbf{q}) \equiv sign(\mathbf{q}) \cdot |\mathbf{q} - \mathbf{p}_i|$. Since any point $\mathbf{q}$ on a triangle can be represented as the linear combination of the three vertex positions $\mathbf{q}_i$, $\mathbf{q}_j$ and $\mathbf{q}_k$, Eqn. (2.11) can be derived by substituting the following equations in the numerator of Eqn. (2.13):

$$\mathbf{q} = \alpha(\mathbf{q}) \cdot \mathbf{q}_i + \beta(\mathbf{q}) \cdot \mathbf{q}_j + \gamma(\mathbf{q}) \cdot \mathbf{q}_k \qquad (\alpha(\mathbf{q}) + \beta(\mathbf{q}) + \gamma(\mathbf{q}) = 1) \tag{2.14}$$

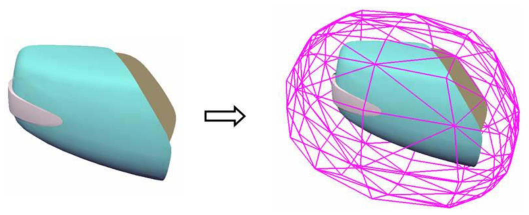
See [14] for an efficient computation of the MVC.

3. FRAMEWORK FOR INTERACTIVE DEFORMATION OF ASSEMBLY MODEL

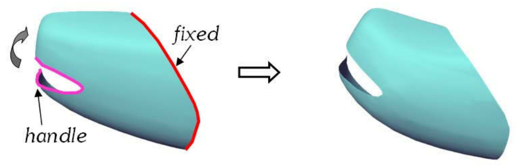
Since feature-preserving deformation cannot propagate the deformation of one model to other disconnected models, we combine feature-preserving deformation and MVC-based deformation. In our framework, the user first selects a target model and deforms it using feature-preserving deformation. The target model that the user directly manipulates is termed the *primary target model*. Other target models are deformed simultaneously and consistently when the user moves the handle region of the primary target model.

Fig. 2 shows a deformation process for an assembly model. First the system calculates the control mesh that encloses the assembly model (Fig. 2(a)). We use a smooth bounding mesh as a control mesh. The MVC of the target models are calculated using this control mesh. Before the assembly model is deformed, the user specifies the positional and feature constraints on each target model, if necessary. The user then selects a primary model and deforms it using feature-preserving deformation (Fig. 2(b)). In the third step, the control mesh is modified using the deformed primary model (Fig. 2(c)). The deformed control mesh is used to deform other target models using MVC-based deformation (Fig. 2(d)). Finally, the deformed assembly model is obtained (Fig. 2(e)).

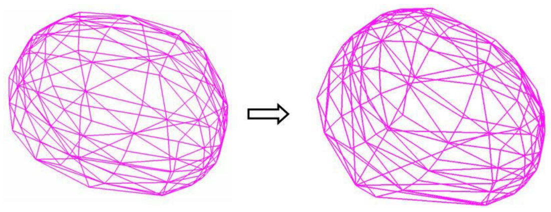
From the user's point of view, the deformation process consists of the pre-processing phase and the interactive phase. In the pre-processing phase, all the linear systems are set up and their matrices are factorized for interactive manipulation. The MVC of each target model are also calculated in this phase. This process normally takes several seconds for mesh models with thousands of vertices. In the interactive phase, the linear systems are quickly solved using factorized matrices. The positions of the vertices are promptly updated when the user drags the handle region of the primary target model on the screen.



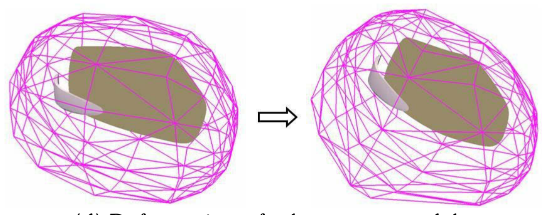
(a) Assembly models and their bounding mesh.



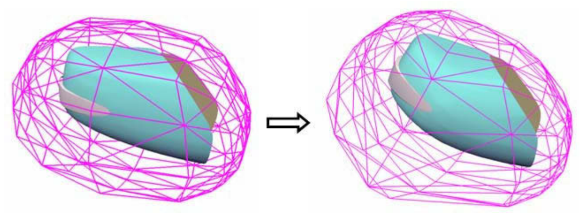
(b) Deformation of a primary mode.



(c) Calculation of the deformed control mesh.



(d) Deformation of other target models.



(e) Deformed assembly model.

Fig. 2: Deformation process of an assembly model.

3.1 Generation of Control Meshes

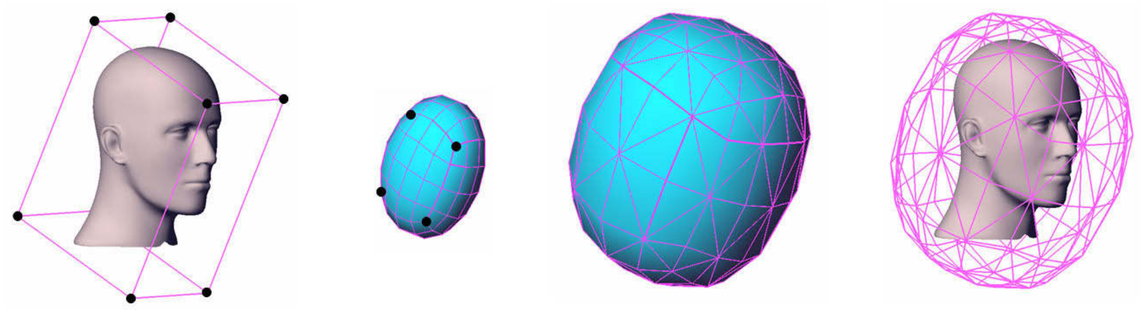
MVC-based deformation requires a control mesh that envelops an assembly model. We generate a smooth control mesh according to the following steps:

(1) First we calculate an oriented bounding box (OBB) [23], which is generated using the center point and three orthogonal principal axes (Fig. 3(a)).
(2) The OBB is subdivided a few times using the Catmull–Clark subdivision scheme (Fig. 3(b)).
(3) Then the vertices of the subdivided mesh is updated by solving:

$$\min\left\{\sum_{i=1}^{n_m}\mathbf{L}(\mathbf{q}_i)^2+\sum_{j=1}^{8}(\mathbf{p}_j-\hat{\mathbf{p}}_j)^2\right\} \tag{3.1}$$

so that the initial eight corner vertices, which are shown by circles in Fig.3(b), are located at the original corner positions of the OBB (Fig. 3(c)). n_m is the number of vertices in the subdivided mesh.

(4) The result is a smooth mesh model that encloses the target models (Fig. 3(d)).



(a) Oriented bounding box. (b) Catmull-Clark subdivision. (c) Constrained smoothing. (d) Control mesh.

Fig. 3: Generation of a smooth control mesh that encloses target models.

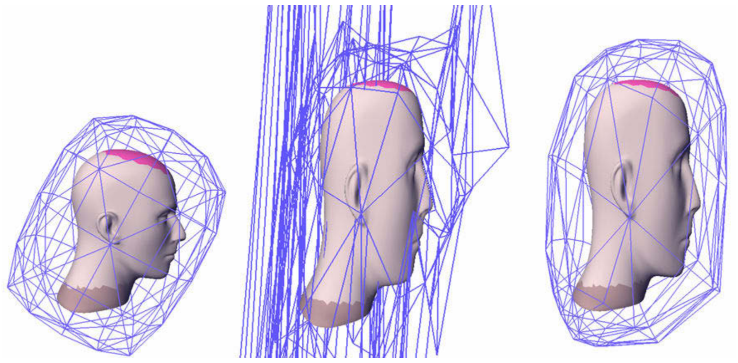
3.2 Deformation of Control Meshes

Mean value coordinates are normally used to deform target models using a control mesh. Instead, we calculate a control mesh using the primary target model. In this case, $\{\mathbf{p}_i\}$ and $\{w_{ij}\}$ in Eqn. (2.11) are given; $\{\mathbf{q}_j\}$ are calculated by solving a linear system. When $\{w_{ij}\}$ $(1\le i\le n;\ 1\le j\le m)$ are represented as the matrix $M_m\in\mathbb{R}^{n\times m}$, Eqn. (2.11) is described as $M_m\mathbf{q}=\mathbf{p}$. Therefore the least squares system is obtained as:

$$M_m^T M_m\mathbf{q}=M_m^T\mathbf{p}\,. \tag{3.2}$$

Unfortunately, this least squares system generates a very noisy mesh. Fig. 4(a) shows a target model and its control mesh. When the target model is deformed, the control mesh is updated by solving Eqn. (2.16). However, the deformed control mesh is too noisy to adequately deform other target models, as shown in Fig. 4(b). One remedy is to prepare a control mesh that tightly envelops target models. Huang et al. [16] calculated such a control mesh for a single connected mesh model using the progressive convex hull construction algorithm [22]. However, it is actually impossible to construct a single connected closed mesh that tightly envelops all the mesh models.

We introduce a different approach for calculating control meshes. For stably computing the least squares system, we solve the following minimization problem:



(a) Original control mesh. (b) Unstable control mesh. (c) Smooth control mesh.

Fig. 4: Stable computation of control meshes.

$$\min\left\{\sum_{i=1}^{n_p}(\mathbf{p}_i - \sum_j w_{ij}\mathbf{q}_j)^2 + cA_m \cdot \frac{n_p}{n_m}\sum_{j=1}^{n_m}\mathbf{L}(\mathbf{q}_j)^2\right\}, \tag{3.3}$$

where n_p and n_m are the numbers of vertices in the primary target model and the control mesh, respectively; A_m is the average of triangle areas in the control mesh; c is a constant value. The value of c was specified as 0.05 for the examples in this paper. Since Eqn. (2.17) penalizes a noisy mesh, it can produce a smooth mesh, as shown in Fig. 4(c).

Since the least squares matrix is a symmetric positive-definite, it can be factorized using the Cholesky factorization [24]. Although the factorized matrices are not sparse in this case, we can solve the linear system interactively, because the number of vertices in the control mesh is relatively small.

4. DEFORMATION OF ASSEMBLY MODELS

In our framework, a primary target model is first deformed using feature-preserving deformation and then the control mesh is updated using the primary model. Other target models are deformed using the updated control mesh. While a single control mesh is used to enclose an assembly model in the previous section, multiple control meshes can also be applied for propagating constraints between disconnected meshes. In this section, we first illustrate the deformation with a single control mesh and then propose a deformation method with multiple local control meshes.

4.1 Deformation with a Single Control Mesh

When the primary model and other target models are enclosed by the same control mesh, they are *similarly* deformed using the control mesh. This method is effective to simultaneously deform multiple target models in similar ways.

Fig. 5 shows a deformation result of the assembly model shown in Fig. 2. When the primary model (the mirror cover) was bent by moving the handle region, the other two target models were consistently and simultaneously deformed. In this example, the primary model has 706 vertices and other models have 1,264 and 500 vertices, respectively. The CPU time for calculating the MVC of three mesh models was 4.21 seconds and the time for setting up and factorizing the matrix of a control mesh was 1.73 seconds. The CPU time for the pre-computation of the feature-preserving deformation was 0.25 seconds. Once these matrices were set up and factorized, three target models could be deformed interactively. The CPU time was measured on a PC with 1.50-GHz Pentium-M and 1 GB of RAM.

Fig. 6 shows two sheet metal panels of an automobile. Parts A and B have 820 and 1,757 vertices, respectively. Their mean value coordinates were calculated in 4.47 seconds, and the matrix of the control mesh was set up and factorized in 1.79 seconds. The pre-computation time for the feature-preserving deformation was 0.33 seconds for Part A and 0.47 seconds for Part B. These two models could be also simultaneously deformed according to the manipulation of the handle in Part A.

Mathematically, the feature-preserving deformation can uniquely determine all the vertex positions of a mesh model if at least one vertex position is specified as a positional constraint. Therefore, when MVC-based deformation determines one or more vertex positions in a target model, feature-preserving deformation can determine all the vertex positions by solving the optimization problem.

For propagating constraints, we prepared three types of deformation. When the target models include feature constraints, the user can select one of the following deformation modes:

Mode 1: All vertex positions in the target model are determined using MVC-based deformation. When no feature constraints are specified on the target model, this mode can be selected.

Mode 2: The user specifies regions that are approximately deformed using MVC-based deformation. The rest of vertex positions are determined using feature-preserving deformation.

Mode 3: The system automatically selects the regions that are deformed using MVC-based deformation from the unconstrained boundary vertices.

4.2 Deformation with Multiple Local Control Meshes

Since disconnected mesh models do not necessarily have similar shapes, a single control mesh may not deform all target models adequately. Since the volume-based deformation basically deforms mesh models inside the volumetric space in similar ways, we introduce multiple control meshes and localize the deformed region for each control mesh.

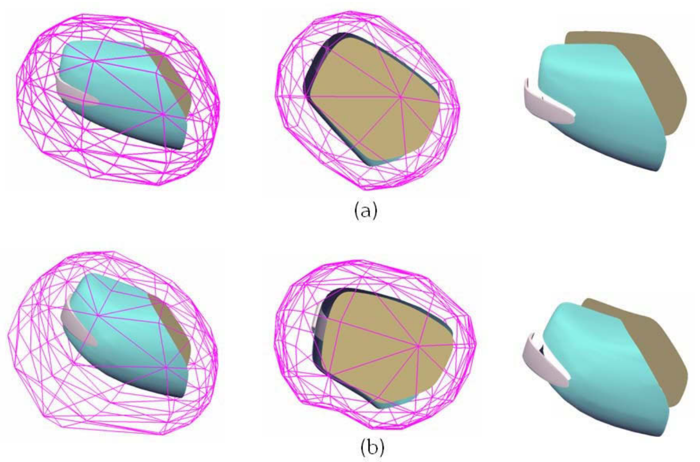
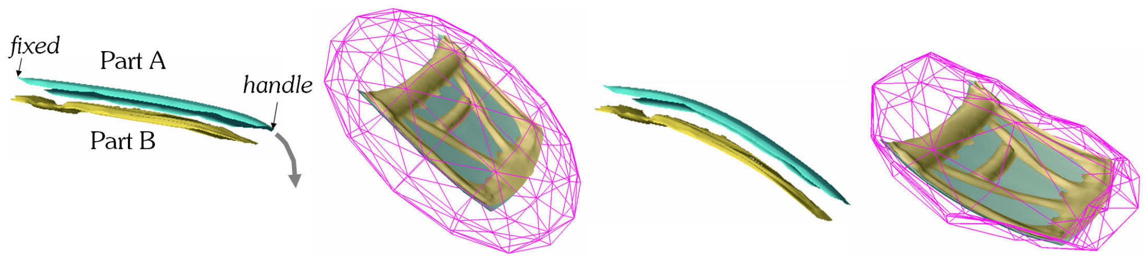


Fig. 5: Deformation results. (a) the original meshes and their control mesh; (b) deformed meshes.



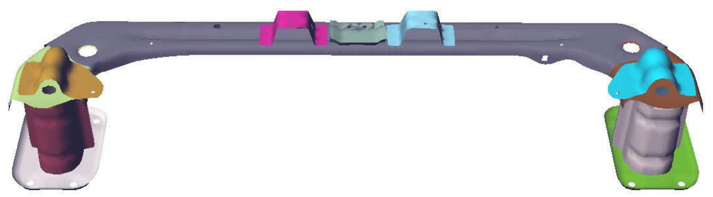
(a) Two target models and their control mesh. (b) Deformed assembly model.

Fig. 6: Deformation result of automobile parts.

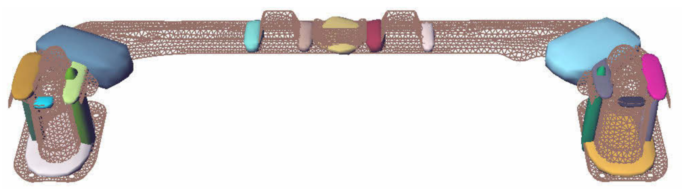
An assembly model is normally constructed by contacting a part with other parts. Then we assume that the contact regions on two disconnected meshes can be deformed using the same control mesh. We introduce local control meshes at contact regions and propagate deformation using the local control meshes.

The deformation process with multiple control meshes is as follows:
(1) The user selects contact regions. Then, the system generates local control meshes and the propagation tree.
(2) The user deforms the primary target model using the feature-preserving deformation.
(3) The target models connected with the primary target model are partly deformed using local control meshes. The deformed vertex positions are used as the positional constraints for the feature-preserving deformation.
(4) The rest of vertex positions are determined using the feature-preserving deformation.
(5) The deformation is propagated to other target models according to the propagation tree.

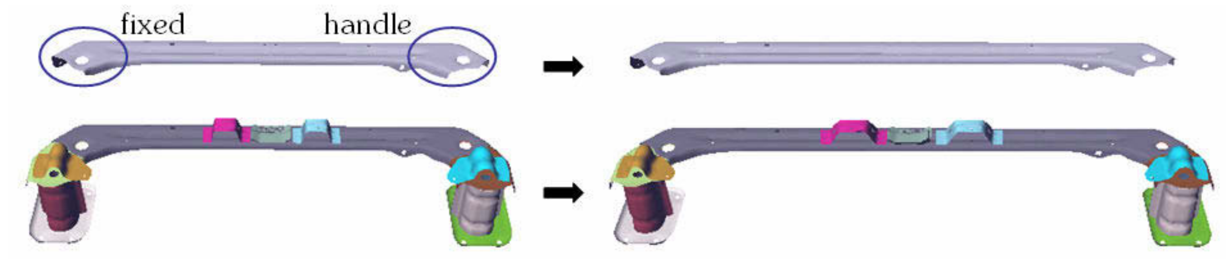
Fig. 7(a) shows an assembly model that consists of 14 disconnected mesh models. This model was designed for CAE analysis and contains 5,963 vertices. To propagate the deformation, contact regions are selected by the user and local control meshes are automatically generated so that each selected region is enclosed by a local control mesh. In Fig. 7(b), we specified 24 contact regions, which are to be bolted or welded. 24 local control meshes were generated using the process described in Section 3.1. When the primary target model is deformed using the feature-preserving deformation, the other target models are simultaneously deformed, as shown in Fig. 7(c). Fig. 7(d) shows the propagation tree of target models. The root of the tree is the primary target model and the children of the root are mesh models connected by local control meshes. In the current implementation, we can select or unselect a region on a mesh model by drawing a 2D closed figure on the screen. We have spent about half an hour for the specification of contact regions.



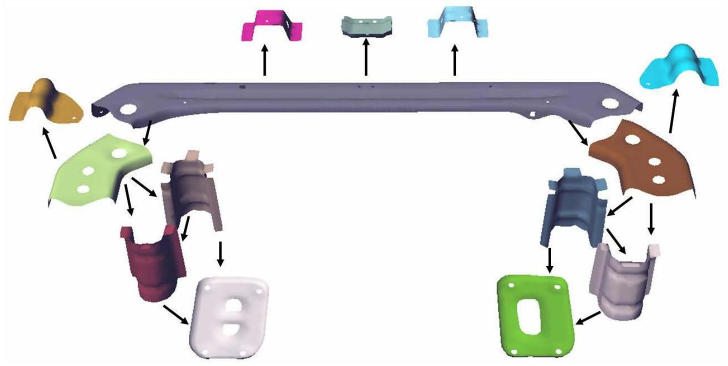
(a) Assembly model with 14 parts.



(b) Local control meshes defined at contact regions.



(c) Deformation of the primary target model (top); deformation of the assembly model (bottom).



(d) Propagation tree based on local control meshes.

Fig. 7: Deformation with local control meshes.

5. CONCLUSION

We proposed a new deformation framework that can simultaneously deform disconnected models while preserving feature constraints. Our method can propagate the deformation of one model to other disconnected models by combining the feature-preserving deformation and the mean value coordinates. In our method, control meshes are interactively deformed using target mesh models. For complicated assembly models, we introduced local control meshes at contact regions and locally propagated the deformation. In our framework, all constraints are represented in linear forms and solved very efficiently using sparse linear system solvers. We showed assembly models could be deformed in interactive rate, followed by practical pre-processing time.

In future work, we would like to investigate the quality problems of deformed meshes. When a mesh model is largely deformed, it may need re-meshing. When a large number of meshes are included in an assembly model, the deformation of those models may not be performed in interactive rate. We would like to improve the performance by incorporating a GPU. In addition, we would like to implement our algorithms on commercial CAD or CAE tools.

6. ACKNOWLEDGEMENTS

3D models of automobile parts in Figs. 2, 5 and 7 are courtesy of MMC. The mannequin model is courtesy of the Graphics and Imaging Laboratory, The University of Washington. The mesh models in Fig. 6 are sample models of the commercial software package Altair/HyperMesh. Kenta Ogawa is thanked for preparing the sample mesh models.

7. REFERENCES

[1] Bloor, M. I. G.; Michael, J. W.: Using partial differential equations to generate free-form surfaces, Computer-Aided Design, 22(4), 1990, 202-212.
[2] Ugail, H.; Bloor, M. I. G.; Michael, J. W.: Techniques for interactive design using the PDE method, ACM Transactions on Graphics, 18(2), 1999, 195-212.
[3] Sorkine, O.; Lipman, Y.; Cohen-Or, D.; Alexa, M.: Rössl, C.; Seidel, H.-P.: Laplacian surface editing, Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, 2004, 175–184.
[4] Botsch, M.; Kobbelt, L.: An intuitive framework for real-time freeform modeling, ACM Transactions on Graphics, 23(3), 2004, 630–634.
[5] Yu, Y.; Zhou, K.; Xu, D.; Shi, X.; Bao, H.; Guo, B.; Shum,H.-Y.: Mesh editing with Poisson-based gradient field manipulation, ACM Transactions on Graphics, 23(3), 2004, 644–651.
[6] Botsch, M.; Bommes, D.; Kobbelt, L.: Efficient linear system solvers for mesh processing. IMA Conference on the Mathematics of Surfaces, 2005, 62–83.
[7] Sederberg, T. W.; Parry, S. R.: Free-form deformation of solid geometric models, Proceedings of SIGGRAPH, 1986, 151–160.
[8] Coquillart, S.: Extended free-form deformation: a sculpturing tool for 3D geometric modeling, Proceedings of SIGGRAPH, 1990, 187–196.
[9] MacCracken, R.; Joy, K. I.: Free-form deformations with lattices of arbitrary topology, Proceedings of SIGGRAPH, 1996, 181–188.
[10] Fontana, M.; Giannini, F.; Meirana, M.: A free-form feature taxonomy, Computer Graphics Forum, 18(3), 1999, 107-118.
[11] Cavendish, J. C.: Integrating feature-based surface design with freeform deformation. Computer-Aided Design, 27(9), 1995, 703-711.
[12] Masuda, H.; Yoshioka, Y.; Furukawa, Y.: Preserving Form-Features in Interactive Mesh Deformation, LNCS 4077 (Geometric Modeling and Processing 2006), 2006, 207-220.
[13] Floater, M. S.: Mean value coordinates, Computer Aided Geometric Design, 20(1), 2003, 19-27.
[14] Ju, T.; Schaefer S.; Warren J.: Mean value coordinates for closed triangular meshes, ACM Transactions on Graphics, 25(3), 2006, 561-566.
[15] Floater, M. S.; Kós, G.; Reimers, M.: Mean value coordinates in 3D, Computer Aided Geometric Design, 22(7), 2005, 623-631.
[16] Huang, J.; Shi, X.; Liu, X.; Zhou, K.; Wei, L.; Teng, S.; Bao, H.; Guo, B.; Shum, H-Y.: Subspace gradient domain mesh deformation, Proceedings of SIGGRAPH, 2006, 1126 – 1134.
[17] Zhou, K.; Huang, J.; Snyder, J.; Liu, X.; Bao, H.; Guo, B.; Shum, H.-Y.: Large mesh deformation using the volumetric graph Laplacian. ACM Transactions on Graphics, 24(3), 2005, 496–503.
[18] Meyer, M.; Desbrun, M.; Schröder, P.; Barr, A. H.: Discrete differential-geometry operators for triangulated 2-manifolds, Visualization and Mathematics III, 2003, 35–57.
[19] Pinkall, U.; Polthier, K.: Computing discrete minimal surfaces and their conjugates, Experimental Mathematics, 2(1), 1993, 15–36.
[20] Alexa, M.: Differential coordinates for local mesh morphing and deformation. The Visual Computer, 19(2-3), 2003, 105–114.
[21] Demmel, J.; Gilbert, J.; Li, X.: SuperLU User's Guide, 1995.
[22] Sander, P. V.; Gu, X.; Gortler, S. J.; Hoppe, H.; Snyder, J.: Silhouette Clipping, Proceedings of SIGGRAPH 2000, 327-334.
[23] Gottschalk, S.; Lin, M.C.; Manocha, D.: OBB Tree: a hierarchical structure for rapid interference detection, ACM Transactions on Graphics, 25(3), 2006, 171-180.
[24] Toledo, S.; Chen, D.; Rotkin, V.: TAUCS: A Library of Sparse Linear Solvers, http://www.tau.ac.il/~stoledo/taucs/, 2003.