Computer-Aided Design and Applications

http://www.cadanda.com

Interactive Deformation of 3D Mesh Models

Hiroshi Masuda[1] and Kenta Ogawa[2]

[1]The University of Tokyo, masuda@nakl.t.u-tokyo.ac.jp
[2]The University of Tokyo, kogawa@nakl.t.u-tokyo.ac.jp

ABSTRACT

This paper proposes a framework for interactively deforming 3D mesh models. 3D mesh models are often used in finite element analysis, but existing surface-based deformation methods fail to deform volumetric elements consistently. In the method proposed here, vertices in a 3D mesh model are classified into boundary vertices and interior vertices. Boundary vertices are treated in the same way as a feature-preserving surface-based deformation; a mean curvature constraint is defined at each boundary vertex, and positional constraints and form-feature constraints are optionally specified at boundary vertices. At each interior vertex, a local volume is constructed as a set of polyhedra that share an interior vertex, and a constraint is assigned based on mean value coordinates. The combination of the mean value coordinates and the mean curvature normals work very well to deform both the boundary and interior of 3D mesh models naturally, without complicated weighting schemes or the rotation of constraints for 3D meshes. In this framework, all constraints are represented in linear forms and solved very efficiently using sparse linear system solvers. This research has shown that 3D mesh models can be deformed in an interactive manner and with a practical amount of preprocessing time.

Keywords: Interactive deformation; 3D mesh; mean curvature; mean value coordinates.
DOI: 10.3722/cadaps.2008.xxx-yyy

1. INTRODUCTION

The recent progress of surface-based deformation enables the interactive deformation of mesh models while preserving various types of constraints. In such techniques, the shape of a mesh model is encoded using differential equations, and the shape is deformed by modifying the boundary conditions of the differential equations [2,7,10–12,15,17–18]. When differential equations are approximated as a linear system, they can be interactively solved using state-of-the-art linear solvers [1]. One appealing feature of surface-based deformation is that the user can specify various types of constraints on vertices, edges, and faces.

One of the most promising applications of surface-based deformation is finite element analysis. In manufacturing industries, many companies rely heavily on CAE analysis to reduce lead time and improve design quality in the early stages of product development. In a trial-and-error optimization process, finite element meshes must be repeatedly modified for structural analysis or collision analysis, but it is very tedious work to modify mesh shapes again and again. It is important for finite element analysis to simplify the mesh editing process. Interactive mesh deformation techniques are very useful for supporting such tasks.

Previous work by the authors [12] proposed a feature-preserving surface-based deformation which precisely preserved the shapes of form-features during deformation. In subsequent work [10,11], the surface-based deformation was extended so that assembly models could be handled by introducing additional constraints between disconnected vertices. In this paper, the feature-preserving deformation is extended to 3D mesh models.

Volume-based deformation such as free-form deformation (FFD) [3,9,14] can deform any shapes, including 3D mesh models, because it deforms geometric shapes by warping the space in which the objects lie. However, volume-based deformation is not intuitive for the user, because manipulation handles do not work directly on geometric shapes. In addition, it is difficult for volume-based deformation to handle engineering constraints such as dimensions or form-features in geometric models.

Figure 1 shows examples of 3D mesh models partly cut off. In deforming 3D mesh models, the following requirements should be satisfied:
(1) Both the boundary and the interior of a 3D mesh are consistently deformed.
(2) Polyhedra are neither reversed nor overlapped during deformation.
(3) The interior structure is preserved as well as possible.
(4) Form-features such as cylindrical holes are preserved during deformation.

Surface-based deformation is suitable for satisfying various types of constraints specified at vertices, edges, and faces on a mesh model. However, this method cannot handle 3D mesh models because it encodes surfaces using the mean curvature, which cannot be defined inside a volume.

Zhou et al. [18] constructed temporal 3D meshes inside a closed 2D mesh for reducing the distortion caused by large deformations of the closed 2D mesh. They introduced a volumetric graph Laplacian which constrains a 3D mesh using Laplacian-like equations. However, they had to use complicated weighting and rotation schemes for the volumetric graph Laplacian. The rotation and weighting schemes strongly affect the quality of the deformed meshes, but they did not discuss the quality of the interior structure of a 3D mesh.

This paper introduces a much simpler deformation method for 3D mesh models. The main contribution of this paper is to propose a set of constraints for consistently deforming the boundary and interior of a 3D mesh model while preserving form-feature constraints. The method proposed here is very simple and easy to implement, requiring neither complicated weighting schemes nor rotation schemes for 3D meshes.

In this method, vertices in a 3D mesh model are classified into boundary vertices and interior vertices, as shown in Fig. 2. Boundary vertices are treated in the same way as feature-preserving surface-based deformations. A mean curvature constraint is defined at each boundary vertex, and positional constraints and form-feature constraints are optionally specified for boundary vertices. At each interior vertex, a local volume is constructed as a set of polyhedra that share the interior vertex. A linear equation is then assigned to each interior vertex, based on mean value coordinates. The combination of the mean value coordinate and the mean curvature normal work very well to deform consistently both the boundary and interior of 3D mesh models.

In the following section, a new constraint is introduced at each interior vertex. Section 3 describes constraints on boundary vertices, and Section 4 describes how to deform 3D mesh models. Section 5 presents experimental results and Section 6 conclusions.

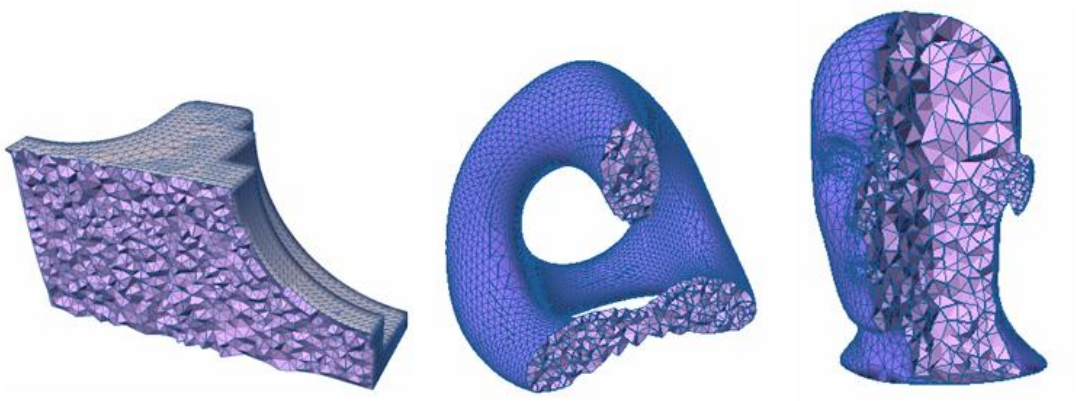


Fig. 1: Examples of 3D mesh models.

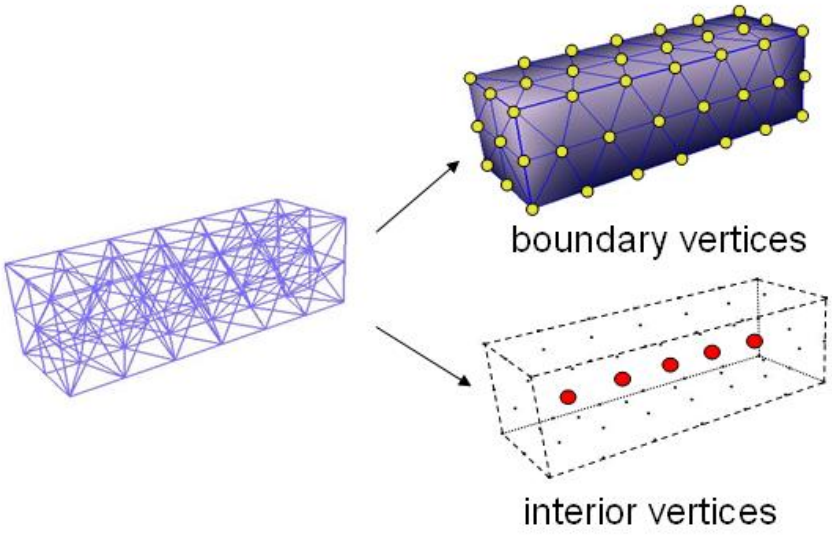


Fig. 2: Boundary vertices and interior vertices of a 3D mesh model.

2. CONSTRAINTS ON INTERIOR VERTICES

Fig. 2 shows the boundary vertices and the interior vertices of a 3D mesh. A vertex is called a *boundary vertex* if it is located on the boundary of a 3D mesh; otherwise it is called an *interior vertex*. At least one equation must be defined at each vertex to determine all vertex positions uniquely. Different types of equations are assigned to boundary and interior vertices. Constraints on interior vertices are described in this section and those on boundary vertices in the next section.

2.1 Mean Value Coordinates

Mean value coordinates are used for defining constraints on interior vertices. They were originally proposed by Floater [5] for smooth interpolation of the inside of a polygon. This idea was extended to 3D space inside a closed triangular mesh by Ju et al. [8] and Floater et al. [6]. Because any polygon can be subdivided into a set of triangles, 3D mean value coordinates can be generally applied to a closed polygonal mesh.

A mean value coordinate in 3D space expresses a point in a closed triangular mesh as a combination of the vertex positions of the closed mesh. A closed mesh for mean value coordinates is called a *control mesh*. Let $(w_1, w_2, ..., w_m)$ be a mean value coordinate. Then point $\mathbf{p} \in \mathbb{R}^3$ inside a control mesh can be represented as the weighted sum of the vertex positions $\{\mathbf{q}_i\}$ of the control mesh:

$$\mathbf{p} = \sum_{i=1}^{m} w_i \mathbf{q}_i \qquad (\sum_{i=1}^{m} w_i = 1) \tag{2.1}$$

Once the mean value coordinate has been calculated, the position $\mathbf{p}$ can be changed by modifying the shape of the control mesh. Because mean value coordinates can achieve a smooth parameterization of the inside of the control mesh, geometric shapes in the control mesh can be naturally deformed [6,8].

Mean value coordinates can be calculated by projecting a control mesh onto a unit sphere. Let $U(\mathbf{p})$ be a unit sphere centered at $\mathbf{p}$, S be a control mesh, and $\tilde{S}$ be the projection of S on $U(\mathbf{p})$. Because the control mesh is a closed surface, $\tilde{S}$ covers the unit sphere. Therefore, the following equation is generally satisfied, because the integral of the unit normal vectors on a sphere is $\mathbf{0}$.

$$\int_{\tilde{S}} sign(\mathbf{x}) \frac{\mathbf{x} - \mathbf{p}}{|\mathbf{x} - \mathbf{p}|} d\tilde{S} = \mathbf{0} \quad , \tag{2.2}$$

where $\mathbf{x}$ is a point on S, $sign(\mathbf{x})$ is the sign of the inner product $(\mathbf{n}, \mathbf{x}-\mathbf{p})$, and $\mathbf{n}$ is the normal vector of S at $\mathbf{x}$. If the surface S consists of a set of triangles $\{T_i\}$, Eqn. (2.2) can be converted to:

$$\mathbf{p} = \int_{\tilde{S}} \frac{\mathbf{x}}{dist(\mathbf{x})} d\tilde{S} \Big/ \int_{\tilde{S}} \frac{1}{dist(\mathbf{x})} d\tilde{S} = \left(\sum_i \int_{\tilde{T}_i} \frac{\mathbf{x}}{dist(\mathbf{x})} d\tilde{T}_i \right) \Big/ \int_{\tilde{S}} \frac{1}{dist(\mathbf{x})} d\tilde{S} \quad , \tag{2.3}$$

where $dist(\mathbf{x}) \equiv sign(\mathbf{x}) \cdot |\mathbf{x}-\mathbf{p}|$. Because any point $\mathbf{x}$ on a triangle can be represented as a linear combination of the three vertex positions $\mathbf{q}_i$, $\mathbf{q}_j$ and $\mathbf{q}_k$, Eqn. (2.1) can be derived by substituting the following equation into the numerator of Eqn. (2.3):

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

See [8] for an efficient computation method for the mean value coordinates.

2.2 Constraints on Interior Vertices

An equation at each interior vertex can be derived using a mean value coordinate. For each interior vertex, a local volume, defined as a set of polyhedra that share the interior vertex, is assumed. Fig. 3 shows an interior vertex and its local volume. As shown in this figure, the interior vertex is located inside the local volume, and the boundary of the local volume is a closed mesh in which the interior vertex lies.

Therefore, a linear equation can be derived by regarding the boundary mesh of a local volume as the control mesh of mean value coordinates. Let $\mathbf{p}_i$ be an interior vertex and $N(i)$ be an index set of the neighboring vertices of $\mathbf{p}_i$. Then the following equation can be derived at each interior vertex:

$$\mathbf{p}_i - \sum_{j \in N(i)} w_{ij} \mathbf{p}_j = 0 \,, \tag{2.4}$$

where $\{w_{ij}\}$ $(j \in N(i))$ is a mean value coordinate.

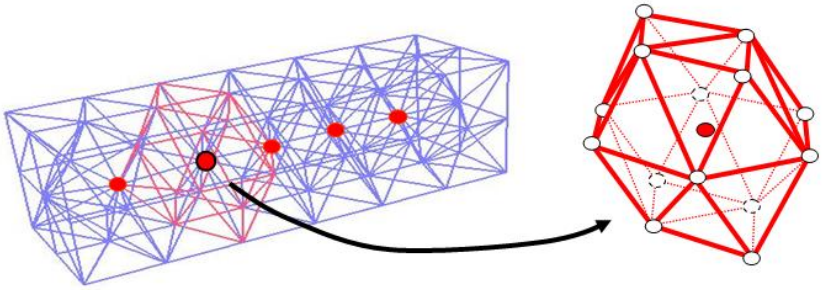


Fig. 3: Local volume of an interior vertex.

To define constraints on interior vertices based on Eqn. (2.4), it is necessary to consider the weights of the constraints. Since the number of constraints is more than the one of variables in our deformation framework, constraints are only approximately satisfied by using the least-squares method and the weight of each constraint determines how well the constraint is satisfied. If equal weights are assigned to all equations, small polyhedra may be overlapped or reversed.

Because smaller local volumes require stronger constraints to avoid an inconsistent mesh structure, we define the following weighted constraints on interior vertices:

$$V(\mathbf{p}_i) \equiv \frac{1}{S_i} \left(\mathbf{p}_i - \sum_{j \in N(i)} w_{ij} \mathbf{p}_j \right) = 0 \qquad (i \in I) \,, \tag{2.5}$$

where S_i is the area of the boundary surface of local volume i, and I is an index set of interior vertices in a 3D mesh. This definition works very well even when a 3D mesh consists of various sizes of polyhedra, as shown in Figs. 6 - 12. It should be noted that Eqn. (2.5) has the same dimension as mean curvature, which is used to constrain boundary vertices.

When the user wants to specify a fixed region in interior vertices, the following constraints are added to internal vertices:

$$\mathbf{p}_i = \mathbf{u}_i \qquad (i \in I_p), \tag{2.6}$$

where $\mathbf{u}_i$ is a user-defined position; I_p is an index set of vertices whose positions are determined by the user.

3. CONSTRAINTS ON BOUNDARY VERTICES

Constraints on boundary vertices are defined in this section. The same constraints are assigned to boundary vertices as in the feature-preserving surface-based deformation proposed by Masuda et al. [12]. Fig. 4 shows examples of feature-preserving deformations. In Fig. 4(c,d), the shapes of circular holes are preserved during deformation, because form-feature constraints are specified at circular holes.

Three types of constraints are specified in feature-preserving deformation: discrete mean curvature, positional constraints, and form-feature constraints.

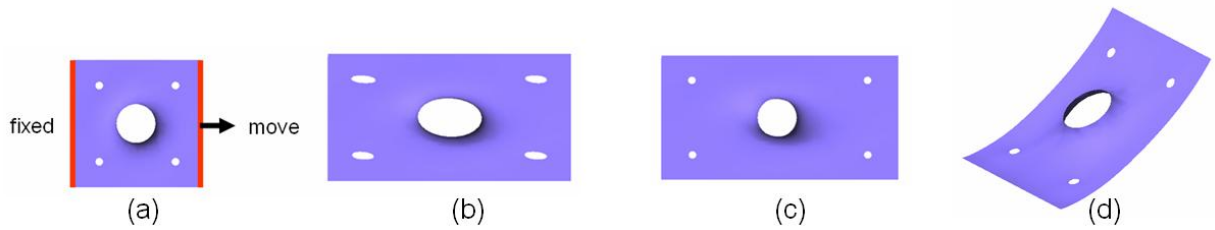


Fig. 4: Deformation of a 2D mesh model.
(a) Original shape; (b) deformed shape with no form-feature constraints; (c) the shapes of circles are preserved by form-feature constraints; (d) five circles are rotated while preserving their shapes.

3.1 Constraints on Positions of Boundary Vertices

Let B be an index set of boundary vertices in a 3D mesh M, and let $\mathbf{P}_b = \{\mathbf{p}_i\}$ ($i \in B$) be the positions of the boundary vertices. Let a boundary mesh M_b be a pair $\{K, \mathbf{P}_b\}$, where K consists of vertices i, edges (i, j), and faces (i, j, k) $(i, j, k \in B)$. The adjacent boundary vertices of vertex i are denoted by $N_b(i) = \{j \mid (i, j) \in K\}$. The original position of $\mathbf{p}_i$ is referred to as $\hat{\mathbf{p}}_i$.

The normal vector and mean curvature on the boundary surface at boundary vertex i are referred to as κ_i and $\mathbf{n}_i$. Then the discrete mean curvature normal, $\kappa_i \mathbf{n}_i$, can be represented as:

$$\kappa_i \mathbf{n}_i = \mathbf{L}(\mathbf{p}_i) \equiv \frac{1}{4A_i} \sum_{j \in N_b(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(\mathbf{p}_i - \mathbf{p}_j), \tag{3.1}$$

where A_i is a Voronoi area on the boundary mesh M_b, and α_{ij} and β_{ij} are the two angles opposite the edge in the two triangles that share edge (i, j), as shown in Fig. 5.

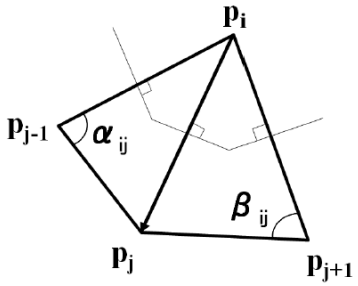


Fig. 5: Definitions of α_{ij} and β_{ij}

If the mean curvature normals of the original mesh are denoted by $\{\delta_1, \delta_2, \ldots, \delta_n\}$, the following constraints can be defined:

$$\mathbf{L}(\mathbf{p}_i) = R_i\delta_i \qquad (i \in B), \tag{3.2}$$

where R_i is the rotation matrix described in section 3.2.

In a typical interactive deformation, the user selects a fixed region, which remains unchanged, and a handle region, which is used as the manipulation handle. Here such constraints are called *positional constraints*. Positional constraints are described as:

$$\mathbf{p}_i = \mathbf{u}_i \qquad (i \in B_p), \tag{3.3}$$

where $\mathbf{u}_i$ is a user-defined position, and B_p is an index set of vertices whose positions are determined by the user.

A form-feature is a partial shape that has an engineering meaning. The shapes of form-feature regions are preserved by:

$$\mathbf{p}_i - \mathbf{p}_j = R_i(\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j) \qquad ((i,j) \in E_f), \tag{3.4}$$

where R_i is a rotation matrix at vertex i, and E_f is a set of edges in the form-feature regions [12]. These equations preserve the relative positions of vertices in the form-feature regions.

3.2 Constraints on Rotations

The rotation matrix R_i in Eqns. (3.2) and (3.4) is determined before vertex positions are calculated. R_i can be represented using rotation axis $\mathbf{v}_i$ and angle θ_i. To interpolate rotations, the rotation can be represented using a unit quaternion by regarding $\mathbf{v}_i$ as three distinct imaginary numbers:

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

The quaternion logarithm:

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

is assigned to vertex i, and the following constraints are introduced for smooth interpolation of rotations:

$$\mathbf{L}(\mathbf{r}_i) = 0 \qquad (i \in B). \tag{3.5}$$

It is possible to specify the user-defined rotation $\mathbf{c}_i$ for vertices to which positional constraints are assigned:

$$\mathbf{r}_i = \mathbf{c}_i \qquad (i \in B_p). \tag{3.6}$$

The same rotations can be specified for vertices in a form-feature region:

$$\mathbf{r}_i - \mathbf{r}_j = 0 \quad ((i,j) \in E_f), \tag{3.7}$$

Because the number of Eqns. (3.5), (3.6), and (3.7) exceeds the number of boundary vertices, rotations are determined using the least-squares method:

$$\arg\min_{\{\mathbf{r}_i\}} \left(\sum_{i \in B} L(\mathbf{r}_i)^2 + \beta \sum_{i \in B_p} (\mathbf{r}_i - \mathbf{c}_i)^2 + \beta \sum_{(i,j) \in E_f} (\mathbf{r}_i - \mathbf{r}_j)^2 \right), \tag{3.8}$$

where β is a constant weight for positional constraints and form-feature constraints. β must be large compared to mean curvature. The rotation matrix R_i in Eqns. (3.2) and (3.4) is determined so that R_i rotates a vector around axis $\mathbf{r}_i / |\mathbf{r}_i|$ by angle $2|\mathbf{r}_i|$.

The Lagrange multiplier is also useful when it is difficult to estimate an adequate value of β. It is possible to solve rotations using the Lagrange multiplier by regarding positional and form-feature constraints as hard constraints [12]:

$$\arg\min_{\{\mathbf{r}_i\}} \left(\sum_{i \in B} L(\mathbf{r}_i)^2 \right) \quad \text{subject to} \quad \begin{cases} \mathbf{r}_i - \mathbf{c}_i = 0 & (i \in B_p) \\ \mathbf{r}_i - \mathbf{r}_j = 0 & ((i,j) \in E_f) \end{cases} \tag{3.9}$$

4. DEFORMATION OF A 3D MESH

For deforming a 3D mesh model, the following constraints are introduced:

$$\begin{cases} V(\mathbf{p}_i) = 0 & (i \in I) & \text{; constraints by mean value coordinates} \\ L(\mathbf{p}_i) = R_i\delta_i & (i \in B) & \text{; constraints by mean curvature normals} \\ \mathbf{p}_i = \mathbf{u}_i & (i \in B_p \text{ or } i \in I_p) & \text{; positional constraints} \\ \mathbf{p}_i - \mathbf{p}_j = R_i(\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j) & ((i,j) \in E_f) & \text{; form-feature constraints} \end{cases} \tag{4.1}$$

In the method presented here, rotations are calculated only for boundary vertices. It is not necessary to perform an explicit rotation of the constraints on interior vertices, because the right-hand side of Eqn. (2.5) is 0. Since reasonable rotation schemes for 3D mesh models are not known, this is an obvious advantage of the present method.

Constraints on boundary vertices and interior vertices can be represented by a single linear system. All vertex positions can be calculated using the least-squares method followed by Eqn. (3.8):

$$\arg\min_{\{\mathbf{p}_i\}} \left(\sum_{i\in I} V(\mathbf{p}_i)^2 + \sum_{i\in B} (L(\mathbf{p}_i) - R_i\delta_i)^2 + \beta \sum_{i\in B_p} (\mathbf{p}_i - \mathbf{u}_i)^2 + \beta \sum_{(i,j)\in E_f} (\mathbf{p}_i - \mathbf{p}_j - R_i(\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j))^2 \right). \tag{4.2}$$

Eqn. (4.2) can be converted to a linear system with a sparse positive-definite symmetric matrix, which can be efficiently solved by Cholesky factorization [16].

It is also possible to solve Eqn. (4.1) using the Lagrange multiplier followed by Eqn. (3.9):

$$\arg\min_{\{\mathbf{p}_i\}} \left(\sum_{i\in I} V(\mathbf{p}_i)^2 + \sum_{i\in B} (L(\mathbf{p}_i) - R_i\delta_i)^2 \right) \quad \text{subject to} \quad \begin{cases} \mathbf{p}_i = \mathbf{u}_i & (i \in B_p) \\ \mathbf{p}_i - \mathbf{p}_j = R_i(\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j) & ((i,j) \in E_f) \end{cases} \tag{4.3}$$

Eqn. (4.3) can be converted to a linear system with a symmetric sparse matrix. Although the matrix is not positive definite, it can be efficiently decomposed by SuperLU [4].

Once the matrices for rotations and positions have been decomposed into products of two triangular matrices, vertex positions can be interactively calculated for different $\mathbf{c}_i$ and $\mathbf{u}_i$ by using forward and backward substitution.

5. EXPERIMENTAL RESULTS

In all examples in this paper, the user specifies a fixed region and a handle region on the boundary of a 3D mesh model. The user can optionally specify form-feature constraints on the boundary of a 3D mesh. Then the system constructs matrices for linear systems and decomposes them before deformation. When matrix decomposition is complete, the user can interactively deform the 3D mesh model by dragging the handle region with a mouse.

The method presented here could successfully deform various 3D mesh models, as shown in Figs. 6-12. Both the boundary and interior of each 3D mesh model were consistently deformed. To visualize the interior, cross-sections were used.

Examples in Figs. 6 and 7 show that our method is effective even when 3D mesh models consist of various sizes of polyhedra. While Fig. 6(a) shows an equally-tessellated 3D mesh, the cat model in Fig. 6(b) has various sizes of tetrahedra. The mannequin model in Fig. 7 has very small polyhedra near eyes and both the boundary and internal vertices in the lower part were fixed. In our method, these models could be consistently and naturally deformed without overlapping or inversion, because our method assigns larger weights on small tetrahedra.

In Figs. 8 – 10, 3D textures were shown to demonstrate the quality of deformed 3D mesh models. In Fig. 8, when the mesh model was stretched, the internal structure was also consistently deformed as well as the boundary. In Fig.9, the lower part of the 3D mesh was fixed. The cross-section views show that the mesh model was consistently deformed while maintaining the fixed part. In Fig. 10, the handle region was rotated. Fig. 10(c) shows that both the boundary and internal meshes were naturally and consistently rotated.

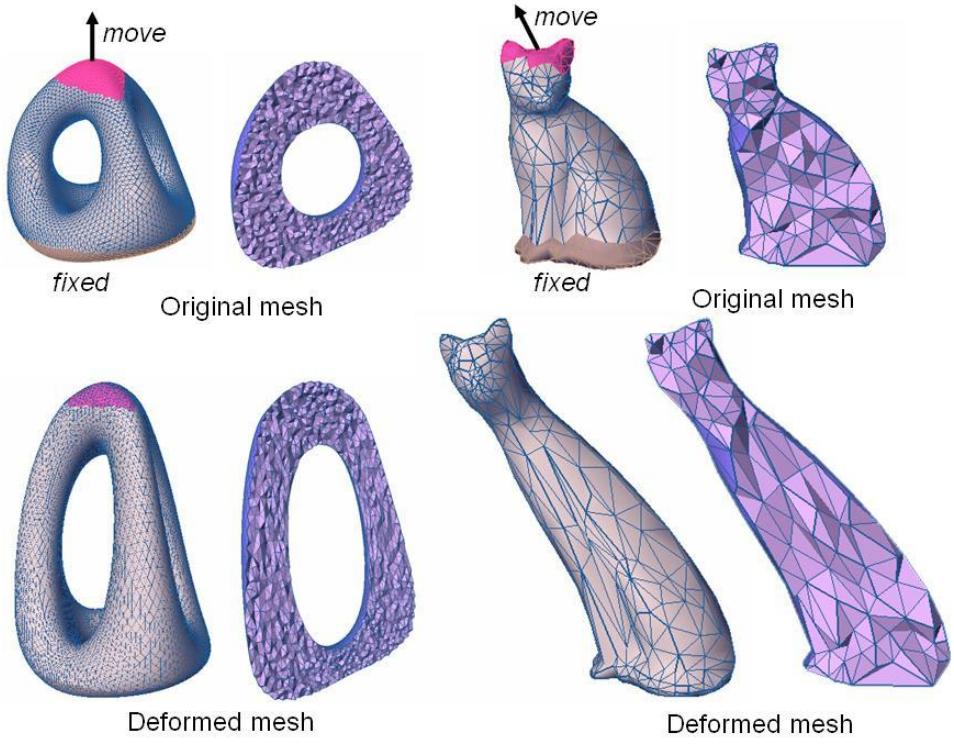
Meshes are often regularly constructed near the boundary surface. The 3D mesh shown in Fig. 11 has strip-like patterns near the smooth surface of the object. Fig. 11 shows that our method could preserve the patterns during deformation.

It is also possible to specify form-feature constraints on the boundary of a 3D mesh model. In Fig. 12, a flat region was constrained using form-feature constraints. The handle region was then moved in two different directions. The flat region was preserved in both cases.

Table 1 shows the computation time for the models presented in this paper. The timing was measured on a laptop PC with a 1.40-GHz Intel Core2 Duo CPU and 2 GB RAM. The least-squares method was used to solve for the constraints. In matrix setup and decomposition, two matrices were processed for rotations and positions. Matrix setup includes the calculation of mean value coordinates for interior vertices. The most time-consuming task was the computation of mean value coordinates.

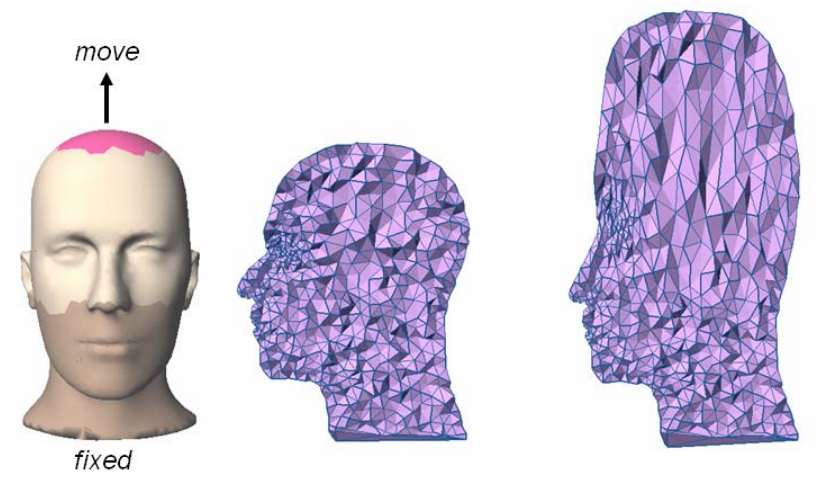
Mesh Model		*Number of Entities*			*CPU Time (sec)*		
Name	*Fig.*	*Tetra*	*V_BDR*	*V_INT*	*Setup*	*Decomp.*	*Total*
cat	Fig. 5(b)	1,828	352	165	0.19	0.03	0.22
part	Fig. 7	13,263	2,666	963	1.47	0.31	1.78
mannequin	Fig. 6	37,456	2,732	5,375	5.25	1.70	6.98
genus3	Fig. 5(a)	93,565	6,652	12,491	15.37	6.09	21.56
fandisk	Fig. 8	94,006	6,475	12,631	16.36	7.69	24.13

Table 1: Computation time for matrix setup and decomposition. Tatra: the number of tetrahedra; V_BDR: the number of boundary vertices; V_INT: the number of interior vertices; Setup: setup of two matrices for rotations and positions; Decomp: decomposing the two matrices.



(a) Genus3 (b) Cat

Fig. 6: Deformed shapes and their sections.



(a) Original mesh (b) Section. (b) Section of deformed mesh.

Fig. 7: Deformation of a partly fixed 3D mesh.

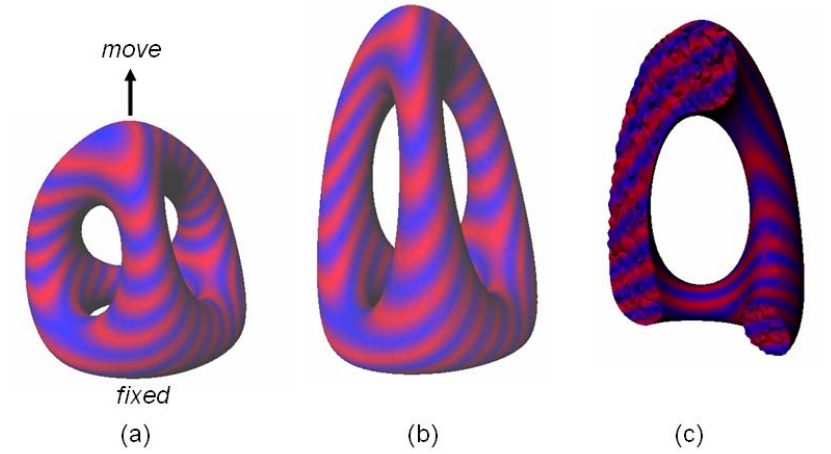


Fig. 8: Mesh quality of deformed model. (a) Original; (b) deformed mesh; (c) section of deformed model.

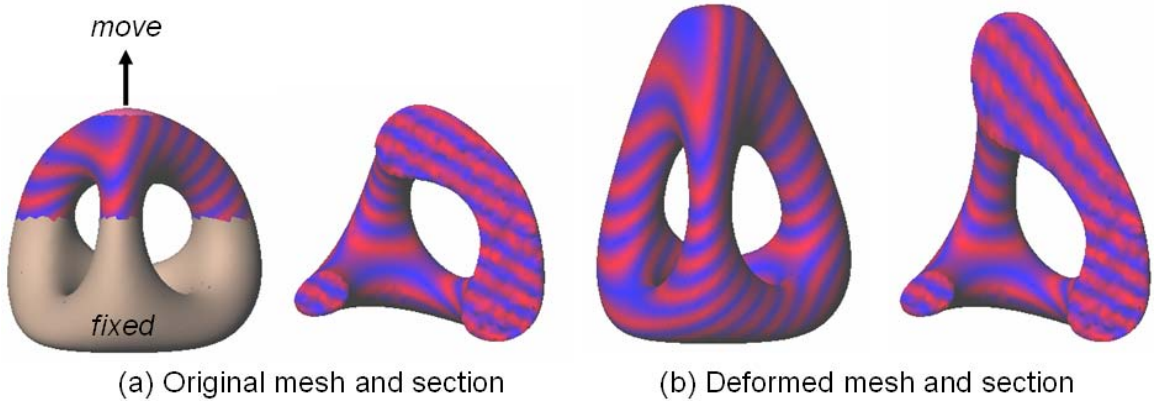


Fig. 9: Quality of a deformed mesh with a large fixed region.

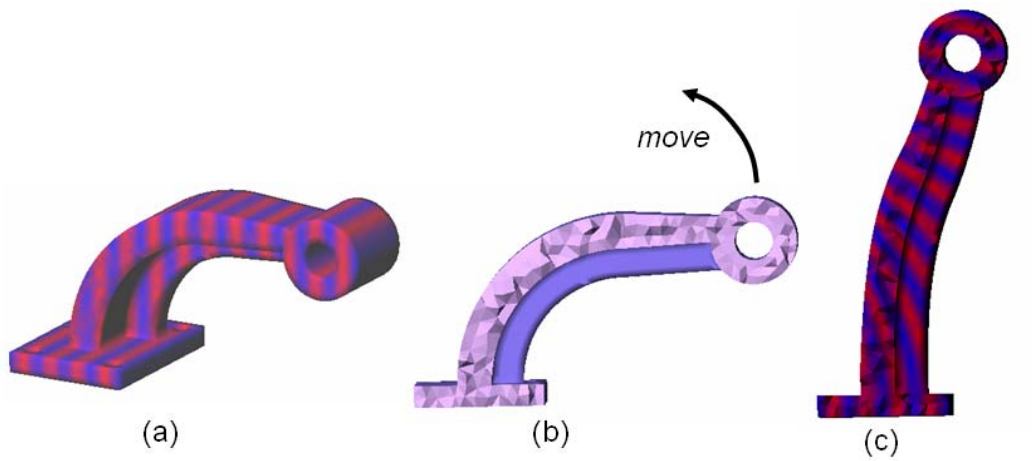
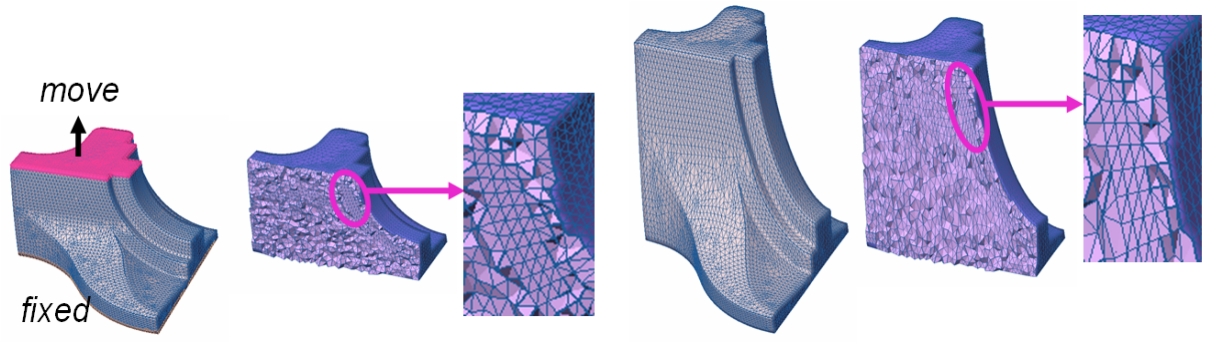


Fig. 10: Mesh quality of rotated mesh. (a) Original; (b) partly-severed model; (c) deformed model with 3D texture.



(a) Original mesh and its cross-section (b) Deformed mesh and its cross-section

Fig. 11: 3D mesh model with regularly triangulated regions.

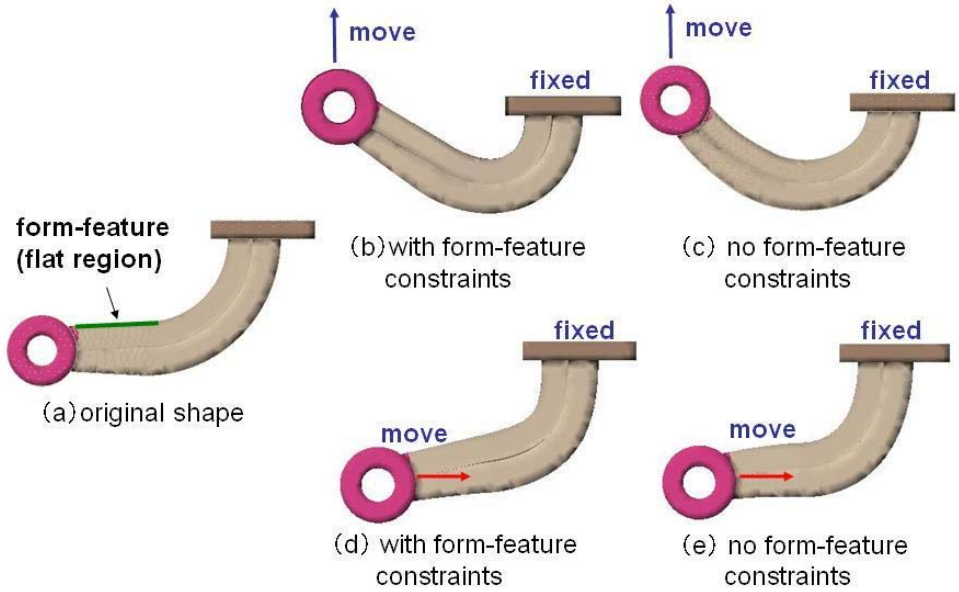


Fig. 12: Feature-preserving deformation of a 3D mesh. (b,c): Moved upward; (d,e): Moved rightward.

6. CONCLUSIONS

We have proposed a new deformation framework for 3D mesh models. Vertices in a 3D mesh were classified into boundary vertices and interior vertices. Three types of constraints were then defined on the boundary vertices: mean curvature constraints, positional constraints, and form-feature constraints. Interior vertices were constrained using mean value coordinates defined in the local volumes. In this framework, all constraints are represented in linear forms and solved very efficiently using sparse linear system solvers. It has been shown that the mean value coordinates and the mean curvature can work very well to deform both the boundary and the interior of a 3D mesh model consistently. In addition, it was shown that 3D mesh models can be deformed in real time and that preprocessing time requirements are reasonable.

Efforts are currently underway to improve the performance of the method using a graphics processing unit (GPU). In the interactive phase, the vertex coordinates are repeatedly updated and transferred to graphics hardware. This is one of the bottlenecks encountered when deforming large mesh models. Matrix computation could possibly be partly performed in GPU without transmitting coordinate data.

REFERENCES

[1] Botsch, M.; Bommes, D.; Kobbelt, L.: Efficient linear system solvers for mesh processing. IMA Conference on the Mathematics of Surfaces, 2005, 62–83.

[2] Botsch, M.; Kobbelt, L.: An intuitive framework for real-time freeform modeling, ACM Transactions on Graphics, 23(3), 2004, 630–634.

[3] Coquillart, S.: Extended free-form deformation: a sculpturing tool for 3D geometric modeling, Proceedings of SIGGRAPH, 1990, 187–196.

[4] Demmel, J.; Gilbert, J.; Li, X.: SuperLU User's Guide, 1995.

[5] Floater, M. S.: Mean value coordinates, Computer Aided Geometric Design, 20(1), 2003, 19-27.

[6] Floater, M. S.; Kós, G.; Reimers, M.: Mean value coordinates in 3D, Computer Aided Geometric Design, 22(7), 2005, 623-631.

[7] 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.

[8] Ju, T.; Schaefer S.; Warren J.: Mean value coordinates for closed triangular meshes, ACM Transactions on Graphics, 25(3), 2006, 561-566.

[9] MacCracken, R.; Joy, K. I.: Free-form deformations with lattices of arbitrary topology, Proceedings of SIGGRAPH, 1996, 181–188.

[10] Masuda, H.; Feature-preserving deformation for assembly models, Computer-Aided Design and Applications, Vol.4, No.1-4, pp.311-320, June 2007.

[11] Masuda, H.; Ogawa, K.: Application of Interactive Deformation to Assembled Mesh Models for CAE Analysis, 2007 ASME International Design Engineering Technical Conferences (IDETC/DAC), Sep., 2007.

[12] Masuda, H.; Yoshioka, Y.; Furukawa, Y.: Preserving Form-Features in Interactive Mesh Deformation, Computer-Aided Design, Vol.39, Issue 5, pp. 361-368, May 2007.

[13] Meyer, M.; Desbrun, M.; Schröder, P.; Barr, A. H.: Discrete differential-geometry operators for triangulated 2-manifolds, Visualization and Mathematics III, 2003, 35–57.

[14] Sederberg, T. W.; Parry, S. R.: Free-form deformation of solid geometric models, Proceedings of SIGGRAPH, 1986, 151–160.

[15] 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.

[16] Toledo, S.; Chen, D.; Rotkin, V.: TAUCS: A Library of Sparse Linear Solvers, http://www.tau.ac.il/~stoledo/taucs/, 2003.

[17] 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.

[18] 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.