Proceedings of the ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2007
September 4-7, 2007, Las Vegas, Nevada, USA

DETC2007-34636

APPLICATION OF INTERACTIVE DEFORMATION TO ASSEMBLED MESH MODELS FOR CAE ANALYSIS

Hiroshi Masuda
masuda@nakl.t.u-tokyo.ac.jp

Kenta Ogawa
kogawa@nakl.t.u-tokyo.ac.jp

School of Engineering
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8856 Japan
Phone/Fax: +081-03-5841-6511

ABSTRACT

Mesh deformation, which is sometimes referred to as mesh morphing in CAE, is useful for providing various shapes of meshes for CAE tools. This paper proposes a new framework for interactively and consistently deforming assembly models of sheet structure for mechanical parts. This framework is based on a surface-based deformation, which calculates the vertex positions so that the mean curvature normal is preserved at each vertex in a least squares sense. While existing surface-based deformation techniques cannot simultaneously deform assembly mesh models, our method allows us to smoothly deform disconnected meshes by propagating the rotations and translations through disconnected vertices. In addition, we extend our deformation technique to handle non-manifold conditions, because shell structure models may include non-manifold edges. We have applied our method to assembly mesh models of automobile parts. Our experimental results have shown that our method requires almost the same pre-processing time as existing methods and can deform practical assembly models interactively.

1. INTRODUCTION

In the manufacturing industry, many companies emphasize on CAE analysis to reduce the lead time and improve the design quality in the early stages of product development. Since engineering products consist of many parts, CAE tools typically process assembly mesh models. To calculate the optimized products, the product shapes are repeatedly modified using CAD tools and evaluated using CAE tools for structural analysis or collision analysis. Such a trial-and-error optimization process is very tedious work. To analyze the assembly mesh models, analysts are forced to edit many parts consistently.

It is important for CAE analysis to simplify the mesh editing processes for assembly models. So far, interactive mesh deformation techniques have been intensively studied [1-11]. Such research aims to develop modeling tools that can intuitively deform mesh models while preserving the details of the shapes. We note that “mesh deformation” is the term used in the field of geometric modeling. The same techniques are typically called “mesh morphing” in CAE. In this paper, we use the term “mesh deformation”, because our technique depends heavily on the contributions developed by geometric modeling community.

There are two typical approaches for interactive mesh deformation; one is volume-based deformation and the other is surface-based deformation.

Volume-based deformation techniques, such as free-form deformation (FFD) [1–3], 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 specified at vertices, edges and faces on a mesh model, because this technique does not directly work on mesh models.

Surface-based mesh deformation encodes geometric shapes using partial differential equations and solves them to determine vertex positions [4–11]. Recently, interactive techniques have been intensively studied for surface-based deformation [6–11]. In such techniques, differential equations are approximated by a sparse linear system and interactively solved. In typical interactive deformation, the user first selects a fixed region, which remains unchanged, and a handle region, which is used as

the manipulation handle. The system decomposes a linear system into the product of upper or lowers triangular matrices. Since the linear system is sparse, this decomposition can be very efficiently computed using state-of-the-art linear solvers [12–15]. Finally, the user interactively deforms the shape by dragging the handle over the screen.

Surface-based deformation is suitable for specifying various constraints at vertices, edges and faces on a mesh model. However, this method cannot propagate deformation to disconnected meshes, because it encodes the shape using the topological connectivity. When applied to an assembly model with disconnected components, it requires tedious manual work to consistently deform the multiple mesh models.

In addition, existing surface-based deformation techniques cannot handle an assembly model that has non-manifold conditions, although the shell structures used in CAE analysis may include non-manifold edges. Non-manifold shells are typically used for simplifying shapes such as ribs in CAE models.

We intend to apply interactive deformation to modify the shapes of the assembly shell models, which are often used to analyze the design of products, such as automobiles [16-18].

Part models in an assembly model often do not *spatially* contact, even if they are bolted or welded. Instead, these parts are *semantically* connected using the attributes of contact conditions, such as beam elements, bolting, contact regions, or offset distances. Figure 1 shows a shell model, in which an offset distance is defined. If two regions are virtually connected by an offset distance, their relative positions should be maintained when the shapes of the mesh models are deformed.

When an assembly model is deformed for CAE analysis, disconnected meshes in the assembly model should be simultaneously and consistently deformed according to the contact conditions. While volume-based deformation can deform disconnected models, it cannot manage the constraints for the contact conditions, which are vital for CAE analysis. Surface-based deformation is suitable for preserving engineering constraints. However, it cannot be applied to disconnected mesh models.

In this paper, we propose a deformation framework that can be applied to assembly models. Our method propagates the contact constraints between disconnected meshes based on surface-based deformation. Our method can also be applied to non-manifold meshes.

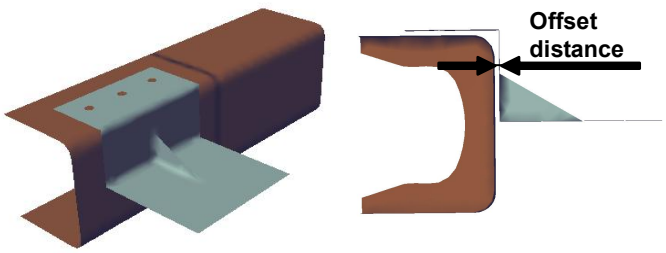


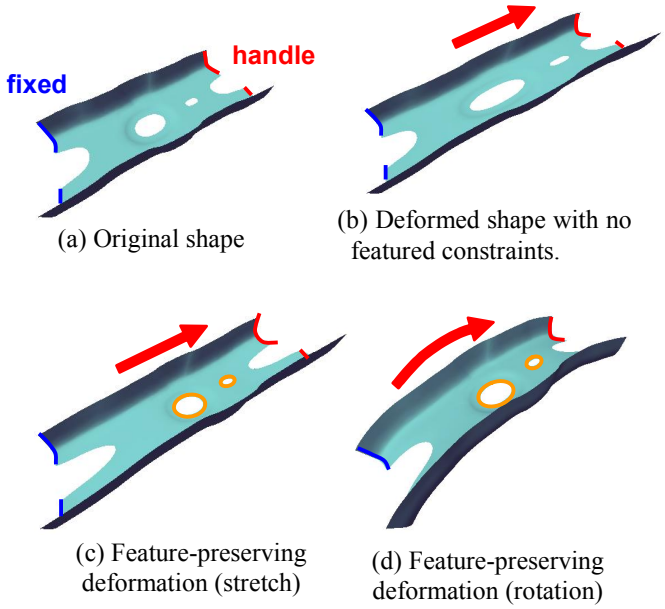
Figure 1. Shell model with an offset distance.

The main contribution of this paper is to propose: (1) a deformation framework that interactively and consistently modifies assembly models of shell structures; and (2) constraints for manipulating mesh models with non-manifold conditions.

In the following section, we describe our surface-based deformation technique described in [11]. In Section 3, we propose our constraint propagation method and constraints for managing non-manifold conditions. In Section 4, we show experimental results. We conclude the paper in Section 5.

2. FEATURE-PRESERVING DEFORMATION

First we explain our feature-preserving deformation method [11]. Figure 2 shows some examples of deformed meshes. The original model is shown in Figure 2(a). In this figure, blue edges are fixed and red edges are treated as a manipulation handle. When the shapes of the holes are not constrained, the original shape is deformed as shown in Figure 2(b). Feature-preserving deformation allows us to preserve the shapes of the form-features during the interactive deformation, as shown in Figure 2(c-d).



(a) Original shape

(b) Deformed shape with no featured constraints.

(c) Feature-preserving deformation (stretch)

(d) Feature-preserving deformation (rotation)

Figure 2. Feature-preserving deformation.

2.1 Constraints for coordinates

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 $N(i) = \{j \mid (i, j) \in K\}$, which is called 1-ring.

The normal vector and mean curvature at vertex i are referred to as κ_i and $\mathbf{n}_i$, respectively. According to [19-21], the discrete mean curvature normal $\kappa_i\mathbf{n}_i$ can be approximated as:

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

where A_i is the Voronoi area, α_{ij} and β_{ij} are the two angles opposite to the edge in the two triangles that share edge (i,j). Since a geometric shape can be encoded using the curvature distribution, the detail shape can be approximately maintained by preserving the original mean curvature in a least-squares sense.

When the mean curvature normals of the original mesh are denoted by $\{\delta_1,\delta_2,\ldots,\delta_n\}$, the error metric of mean curvature is defined as:

$$\sum_{i\in\mathbf{\Lambda}}(\mathbf{L}(\mathbf{p}_i)-R_i\delta_i)^2 \qquad (2)$$

where R_i is the rotation matrix for the normal vector $\mathbf{n}_i$; $\mathbf{\Lambda}$ is the index set of vertices in the mesh model. $\{R_i\}$ $(i\in\mathbf{\Lambda})$ are calculated at all the vertices before the coordinates of the vertices are calculated, as described in the next section.

When the user specifies fixed or handle regions on a mesh, the following *positional constraints* are added as the boundary conditions of differential equations:

$$\mathbf{p}_i = \mathbf{u}_i \quad (i\in\mathbf{\Lambda}_p) \qquad (3)$$

where $\mathbf{u}_i$ is any point that the user specifies; $\mathbf{\Lambda}_p$ is the index set of vertices to which positional constraints are assigned.

We define a form-feature as a partial shape that has an engineering meaning and represent it as a subset of $(K,\mathbf{P})$. We constrain the relative positions of vertices in the form-feature f using the following constraints:

$$\mathbf{p}_i-\mathbf{p}_j = R_i\left(\tilde{\mathbf{p}}_i-\tilde{\mathbf{p}}_j\right) \quad (i,j)\in F_k \quad (k=1,2,\ldots,n_f) \qquad (4)$$

where $\tilde{\mathbf{p}}_i$ and $\tilde{\mathbf{p}}_j$ are coordinates of the original model; R_i $(=R_j)$ 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.

2.2 Constraints for rotations

The rotation matrix R_i in (2) and (4) can be determined by the rotation axis $\mathbf{v}_i$ and rotation angle θ_i at vertex i. A combination of $\mathbf{v}_i$ and θ_i can be also represented using a unit quaternion as:

$$Q_i = \cos\frac{\theta}{2}+\mathbf{v}_i\sin\frac{\theta}{2} = \exp\frac{\theta}{2}\mathbf{v}_i \qquad (5)$$

where rotation axis $\mathbf{v}_i$ is regarded as three distinct imaginary numbers of a quaternion.

Since the logarithm of a unit quaternion is defined as the inverse of the exponential as:

$$\mathbf{r}_i = \ln Q_i = \frac{\theta}{2}\mathbf{v}_i \ . \qquad (6)$$

We assign the quaternion logarithm $\mathbf{r}_i$ to each vertex. A quaternion logarithm assigned to vertex i is referred to as $\mathbf{r}_i$. When the value of $\mathbf{r}_i$ is determined as $\mathbf{c}_i\in\mathrm{R}^3$, the mean curvature normal at vertex i is rotated around axis $\mathbf{c}_i/|\mathbf{c}_i|$ by angle $2|\mathbf{c}_i|$.

For positional constraints and feature constraints, the following two constraints are added for rotations:

$$\begin{cases}\mathbf{r}_j=\mathbf{c}_j & (j=1,2,\ldots,n_p)\\ \mathbf{r}_i-\mathbf{r}_j=0 \quad (i,j)\in F_k & (k=1,2,\ldots,n_f)\end{cases} \qquad (7)$$

where $\mathbf{c}_j$ is a user-defined rotation. The second equations assign the same rotations to vertices in a form-feature. For smoothly interpolating rotations for unconstrained vertices, we introduce the following equations:

$$\mathbf{L}(\mathbf{r}_i)=0 \quad (i\in\mathbf{\Lambda}) \ . \qquad (8)$$

Equation (8) means that the quaternion logarithms $\{\mathbf{r}_i\}$ are smoothly determined so that the surface constructed by $\{\mathbf{r}_i\}$ has zero mean curvature at all the vertices.

2.3 Optimization

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

$$\min\left(\sum_{i\in\mathbf{\Lambda}}\left\|\mathbf{L}(\mathbf{r}_i)\right\|^2+\sum_{j\in\mathbf{\Lambda}_p}w_j\left\|\mathbf{r}_j-\mathbf{c}_j\right\|^2+\sum_{k=1}^{n_f}\sum_{(i,j)\in F_k}w_k'\left\|\mathbf{r}_i-\mathbf{r}_j\right\|^2\right) \qquad (9)$$

$$\min\left(\sum_{i\in\mathbf{\Lambda}}\left\|\mathbf{L}(\mathbf{p}_i)-\mathbf{R}_i\delta_i\right\|^2+\sum_{j\in\mathbf{\Lambda}_p}w_j\left\|\mathbf{p}_j-\mathbf{u}_j\right\|^2+\sum_{k=1}^{n_f}\sum_{(i,j)\in F_k}w_k'\left\|(\mathbf{p}_i-\mathbf{p}_j)-R_i(\tilde{\mathbf{p}}_i-\tilde{\mathbf{p}}_j)\right\|^2\right) \qquad (10)$$

where w_i and w_i' are weight values.

Each of (9) and (10) is partially differentiated with respect to $\{\mathbf{r}_i\}$ and $\{\mathbf{p}_i\}$, and then converted to a sparse linear system, which can be efficiently factorized using state-of-the-art linear solvers [12–14]. Once the matrix is factorized, vertex positions can be interactively solved according to the positions and rotations of the handle region.

3. DEFORMATION FOR ASSEMBLY MODELS

We extend our framework to deform an assembly model for a shell structure, which is used for CAE analysis. In our method, the pairs of vertices are selected from disconnected parts and they are constrained so that the deformation of one part can be propagated to other disconnected parts. We also introduce constraints for managing non-manifold conditions, which often appear in shell structures. Non-manifold edges are deformed so that the angles of the adjacent faces are preserved in a least squares sense. These two types of constraints can be represented as linear equations and solved interactively using factorized matrices.

3.1 Constraints for disconnected vertices

Our method deforms disconnected meshes by adding constraints between pairs of vertices. Figure 3 illustrates the constraints between disconnected vertices. $\mathbf{p}_i^{(1)}$ and $\mathbf{p}_j^{(2)}$ are the positions of vertex i in Mesh-1 and vertex j in Mesh-2, respectively. $\mathbf{r}_i^{(1)}$ and $\mathbf{r}_j^{(2)}$ are quaternion logarithms, which represent the rotations at the vertices. We define a virtual link between these vertices. When the positions and rotations of the vertices are modified in Mesh-1, the deformation is propagated to Mesh-2 through this link.

We formalize the link between two disconnected vertices using the following equations:

$$\begin{cases} (\mathbf{p}_i^{(1)} - \mathbf{p}_j^{(2)}) - R_i^{(1)}\gamma_{i,j} = 0 \\ \mathbf{r}_i^{(1)} - \mathbf{r}_j^{(2)} = 0 \end{cases} \tag{11}$$

where $\gamma_{i,j}$ is the initial value of $(\mathbf{p}_i^{(1)} - \mathbf{p}_j^{(2)})$; $R_i^{(1)}$ is a rotation matrix calculated using $\mathbf{r}_i^{(1)}$ and $\mathbf{r}_j^{(2)}$. Equation (11) maintains the relative positions and angles between two vertices. These constraints are added to (9) and (10).

Figure 4 shows an example of the deformed disconnected meshes. This model consists of 16 parts. 15 links are defined between the nearest points in a chain of primitives. This figure shows that our method can smoothly deform disconnected parts.

We note that we actually solve equations defined in 2-ring vertices, because Equation (1), which is defined in 1-ring vertices, is converted to an equation in 2-ring vertices when the least squares method is applied to solve the equations. This is the reason why deformation can be smoothly propagated through the disconnected vertices, although (11) does not constrain the smoothness of two disconnected vertices

In some cases, it may be convenient to define the constraints between a vertex and an edge, as shown in Figure 5. Then, we specify similar constraints between a vertex and an edge. Since a point on an edge can be represented as $t\mathbf{p}_j^{(2)} + (1-t)\mathbf{p}_k^{(2)}$ using parameter t, the following constraints can be defined between a vertex in Mesh-1 and an edge in Mesh-2:

$$\begin{cases} \mathbf{p}_i^{(1)} - \{t\ \mathbf{p}_j^{(2)} + (1-t)\ \mathbf{p}_k^{(2)}\} - R_i^{(1)}\gamma_i = 0 \\ \mathbf{r}_i^{(1)} - \{t\ \mathbf{r}_j^{(2)} + (1-t)\ \mathbf{r}_k^{(2)}\} = 0 \end{cases} \tag{12}$$

Similarly, we can define constraints between a vertex and a face, because a point on a triangle can be represented as:

$$s_l\mathbf{p}_i + t_l\mathbf{p}_j + u_l\mathbf{p}_k = \mathbf{u}_l \quad (s_l + t_l + u_l = 1)\,. \tag{13}$$

For specifying constraints between two disconnected meshes, the user specifies regions to be constrained, and then the system calculates the distances from each specified vertex to all vertices in other disconnected meshes. When the distance to the nearest point is less than a certain threshold, constraints are specified between the pair entities.

In our experiment, when two or more constraints are specified within 1-ring vertices, the constrained region becomes too rigid. Therefore, we avoid selecting vertex pairs in 1-ring, as shown in Figure 3.

Figure 6 shows examples of deformed meshes. In Figure 6(a), pairs of vertices are constrained in disconnected boundaries, which are shown in green. In Figure 6(b), two mesh models are deformed consistently. In Figure 6(c), different pairs of vertices are connected by links, as shown in green lines. Figure 6(d) shows the deformed shapes. These results show that our method is useful to deform disconnected meshes consistently.

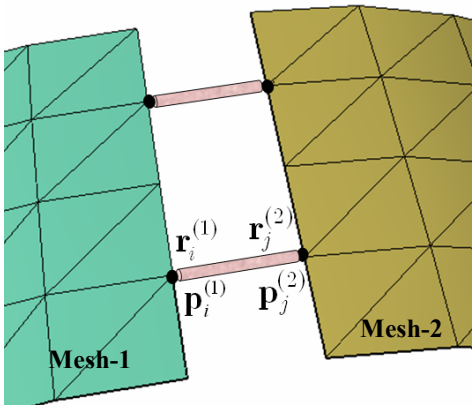


Figure 3. Pair of vertices to be connected.

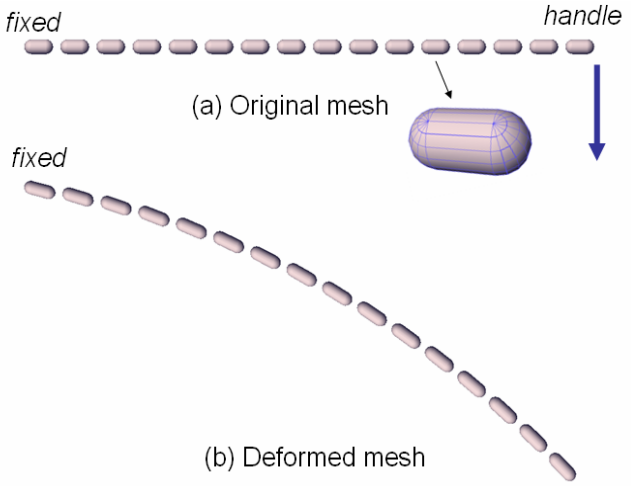


Figure 4. Deformation of disconnected meshes.

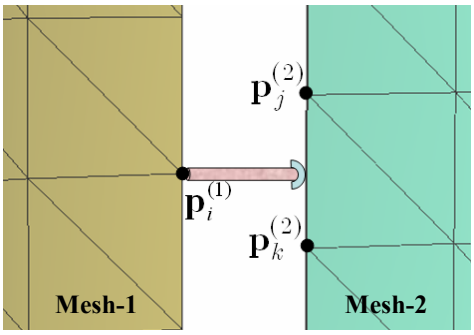
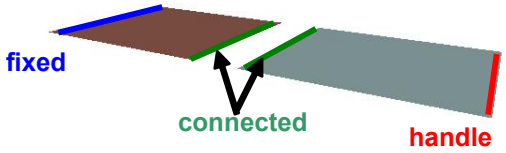
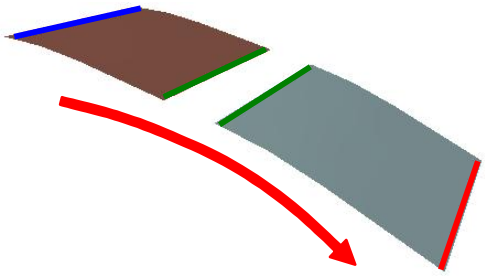


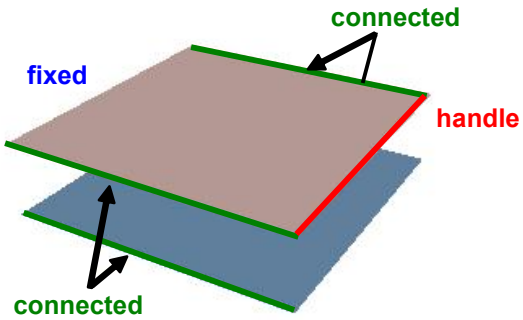
Figure5. Constraint between a vertex and an edge.



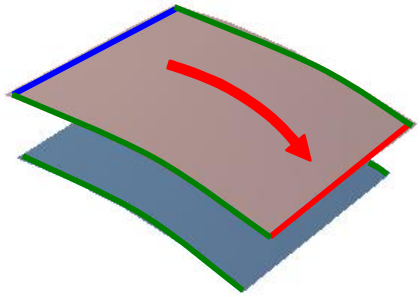
(a) Pair of boundaries is connected.



(b) Deformed shapes.



(c) Two pairs of boundaries are connected.



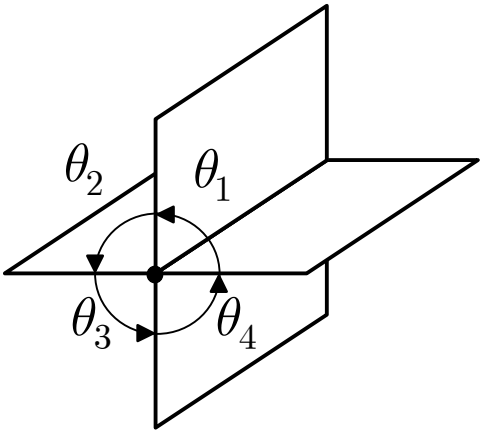
(d) Deformed shape.

Figure 6. Deformation of disconnected planes.

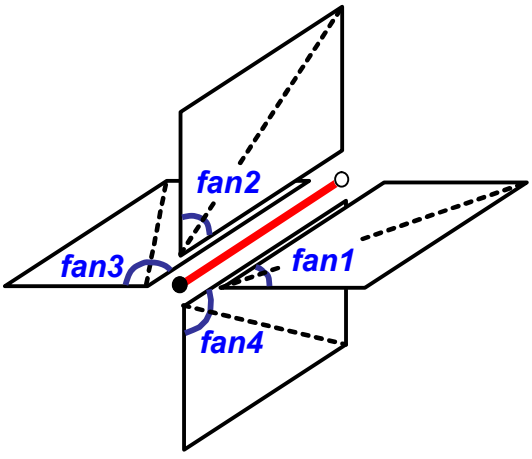
3.2 Constraints for non-manifold conditions

In CAE analysis, the sheet structure may have non-manifold conditions. For example, when a rib is approximated by a sheet, the junction edges are shared by three faces. Since the neighborhood of a non-manifold vertex is not homeomorphic to a plane or a half-plane, its mean curvature normal cannot be defined by (1).

We now introduce the constraints for the non-manifold conditions. We subdivide the faces around a non-manifold vertex into a set of *fan*s, as shown in Figure 7. A fan is defined as a set of connected faces that are homeomorphic to a half-plane. The neighborhood around a vertex in Figure 7(a) is subdivided into four fans, as shown in Figure 7(b).



(a) Non-manifold model.



(b) Four fans.

Figure 7. Non-manifold model and fans.

We introduce the following equation for the fan in Figure 8:

$$L_h(\mathbf{p}_0,m)=\frac{1}{4A(\mathbf{p}_0,m)}\{\sum_{i=2}^{n-1}(\cot\alpha_i+\cot\beta_i)(\mathbf{p}_i-\mathbf{p}_0) + \cot\alpha_n(\mathbf{p}_n-\mathbf{p}_0)+\cot\beta_1(\mathbf{p}_1-\mathbf{p}_0)\} \quad (14)$$

where m represents the mth fan; $A(\mathbf{p}_0,m)$ is the Voronoi area of the mth fan. Each fan is constrained separately.

In Figure 7(a), an edge is shared by four faces. When this model is deformed, the following constraints preserve angles $\{\theta_1,\theta_2,\theta_3,\theta_4\}$:

$$\begin{cases}\dfrac{1}{A(\mathbf{r}_i,m)}L_h(\mathbf{r}_i,m)=0\\ \dfrac{1}{A(\mathbf{p}_i,m)}L_h(\mathbf{p}_i,m)=R_i\delta_i^m\end{cases}\quad(m=1,2,3,4) \quad (15)$$

where δ_i^m is the initial value of (14) for the mth fan. Since the four fans share the rotation matrix R_i, the angles between adjacent fans are preserved.

Figure 9 shows a deformed shape of a non-manifold mesh. This result shows that the deformation is smoothly propagated through the non-manifold edges and the angles between adjacent faces are preserved.

3.3 Implementation

We implemented our framework using C++ on PCs. Mesh models are constructed based on half-edge structure. The graphic user interface is implemented using a cross-platform library FOX toolkit (http://www.fox-toolkit.org), which runs on both Linux and Windows operating systems.

4. EXPERIMENTAL RESULTS

We evaluated our method using shell structure models of automobile parts. In our experiments, assembly models could be deformed interactively, once the matrices were factorized.

The assembly model in Figure 10(a) consists of Part-A and Part-B, each of which has non-manifold edges at the junctions of the ribs. This assembly model has 5,243 faces, 15,729 edges and 2,758 vertices. There is a gap between the two meshes and they do not contact spatially at any point. In Figure 10(b), a fixed region and a handle region are shown by blue and red lines, respectively. We selected the region to be connected in Part-B. The system then detected the nearest vertices in Part-A and added the constraints between the pairs of vertices.

Figure 10(c) shows a deformed shape when the handle region is rotated and translated. When Part-A was deformed, Part-B was also consistently deformed. In Figure 10(d), vertex pairs were selected only in the left side of Part-B. The width of Part-B was preserved when Part-A was stretched. In the both cases, the non-manifold edges are consistently deformed.

Figure 11(a) is an assembly model created for evaluating collision absorption. Comp-A has 4 shell parts, 5,092 faces, 15,276 edges and 2,807 vertices. Comp-B has 10 shell parts, 18,176 faces, 54,528 edges and 9,749 vertexes.

Figure 12 shows the deformed shape of Comp-A. In Figure 12(a), the four holes shown in blue are fixed and the edges in red are specified as a manipulation handle. In this example, we specified connected regions on the assembly model, and the system automatically detected the nearest vertices and the specified constraints between the pairs. Figure 12 (b) shows a deformed shape when the handle was moved upwards.

Figure 13 shows the deformed shape of Comp-B. We fixed the regions shown in blue boxes and specified a handle region as shown in a red box. We repeatedly selected two disconnected meshes to be connected and specified the contact regions. Figure 13(b) shows a deformed shape. This result shows that our method can be applied to relatively complex assembly models.

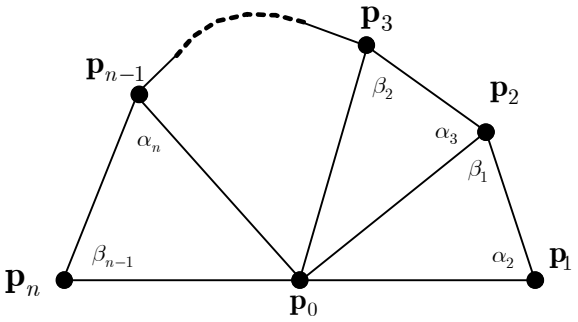
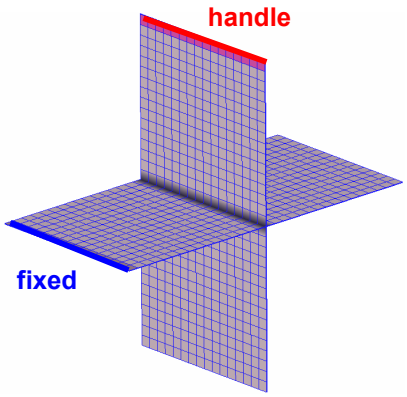
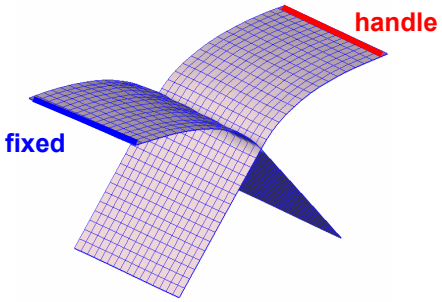


Figure 8. A fan around vertex $\mathbf{p}_0$.

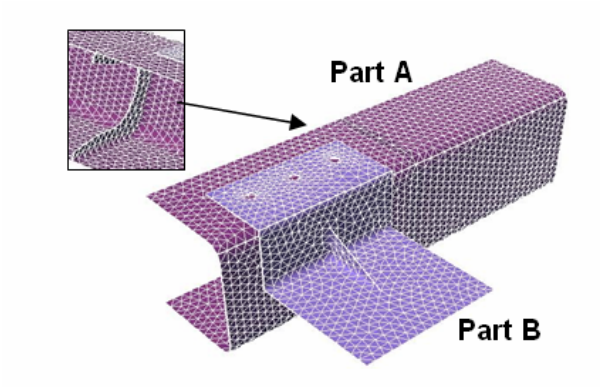


(a) Original model

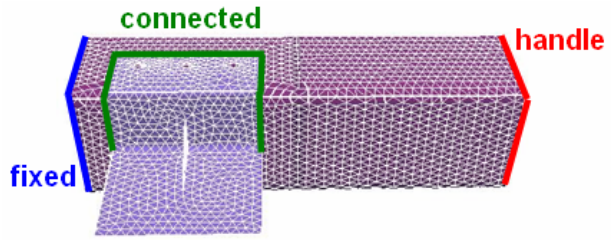


(b) Deformed model

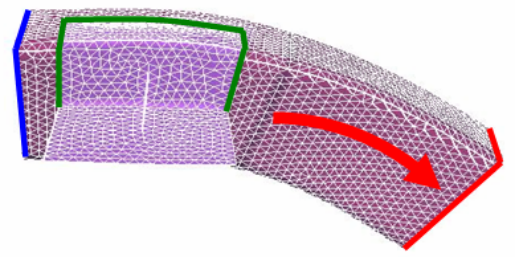
Figure 9. Deformation of non-manifold model.



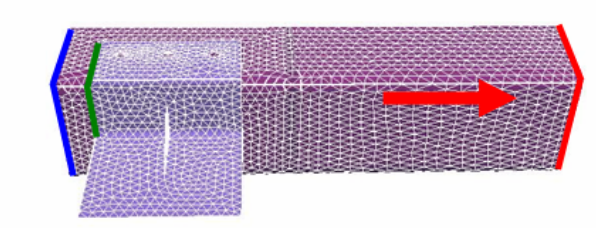
(a) Original meshes.



(b) Constrained regions.

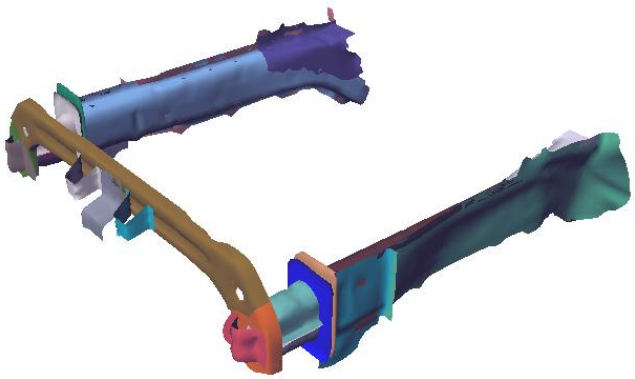


(c) Deformed meshes.

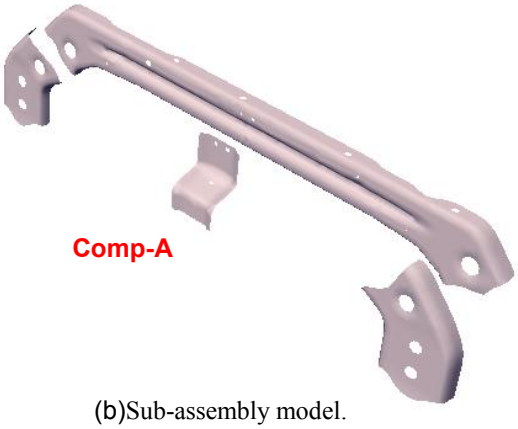


(d) Deformed meshes with different constraints.

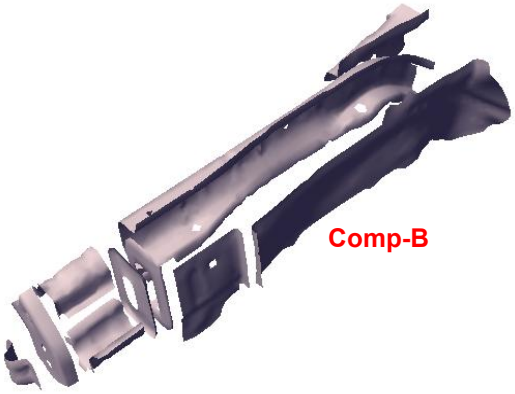
Figure 10. Deformation of an assembly model with non-manifold edges.



(a) Original assembly model

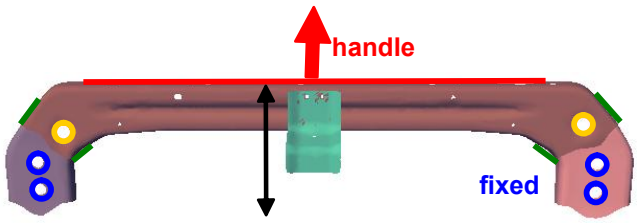


(b)Sub-assembly model.

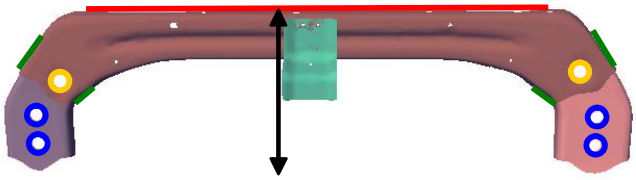


(c) Sub-assembly model.

Figure 11. Assembly models for the evaluation of collision absorption.

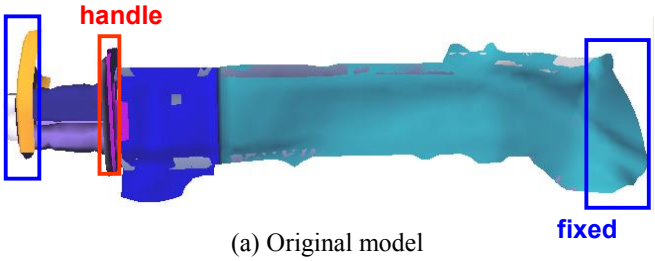


(a) Original model

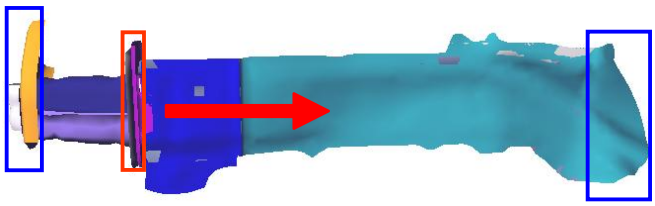


(b) Deformed model

Figure 12. Deformation of Comp-A.



(a) Original model



(b) Deformed model

Figure13. Deformation of Comp-B.

5. CONCLUSION

We have proposed a framework that can interactively and consistently deform disconnected mesh models for shell structures. Our method can propagate the deformation of one mesh to other disconnected meshes by introducing linear constraints between two vertices in disconnected meshes. We have extended our method and enabled to handle non-manifold conditions in shell structure models. In our framework, all the constraints are represented in linear forms and are solved very efficiently using sparse linear system solvers. We have shown that practical assembly models could be consistently and interactively deformed.

In future work, we would like to investigate methods for managing various types of contact attributes used in CAE models. Such information would be useful for detecting vertex pairs effectively and robustly. We also would like to improve the performance of the deformation. Practical assembly models may have hundreds of thousands of vertices. We would like to improve the performance by incorporating GPU (Graphics Processing Unit). Finally, we would like to implement our algorithms on commercial CAD or CAE tools.

ACKNOWLEDGMENTS

The mesh models of automobile parts in Figure 2 and 11–13 were provided courtesy of Mitsubishi Motors Corporation. The mesh models in Figure 1 and 10 are sample models of the commercial software package Altair/HyperMesh.

REFERENCES

[1] Sederberg, T. W. and Parry, S. R, 1986, “Free-Form Deformation of Solid Geometric Models”, *Proceedings of SIGGRAPH 1986*, pp. 151–160.

[2] Coquillart, S., 1990, “Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling”, *Proceedings of SIGGRAPH 1990*, pp. 187–196.

[3] MacCracken, R. and Joy, K. I., 1996, “Free-Form Deformations with Lattices of Arbitrary Topology”, *Proceedings of SIGGRAPH 1996*, pp. 181–188.

[4] Bloor, M. I. G. and Michael, J. W., 1990, “Using Partial Differential Equations to Generate Free-Form Surfaces”, *Computer-Aided Design*, Vol. 22, No. 4, pp. 202-212.

[5] Ugail, H., Bloor, M. I. G. and Michael, J. W., 1999, “Techniques for Interactive Design Using the PDE Method”, *ACM Transactions on Graphics*, Vol. 18, No. 2, pp. 195-212.

[6] Sorkine, O., Lipman, Y., Cohen-Or, D., Alexa, M., Rössl, C. and Seidel, H.-P., 2004, “Laplacian Surface Editing”, *Proceedings of the 2004 Eurographics Symposium on Geometry Processing*, pp. 175–184.

[7] Botsch, M. and Kobbelt, L., 2004, “An Intuitive Framework for Real-Time Freeform Modeling”, *ACM Transactions on Graphics*, Vol. 23, No. 3, pp. 630–634.

[8] Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B. and Shum,H.-Y., 2004, "Mesh Editing with Poisson-Based Gradient Field Manipulation", *ACM Transactions on Graphics*, Vol. 23, No.3, pp. 644–651.

[9] Zayer, R., Rössl, C., Karni, Z. and Seidel, H.-P., 2005, "Harmonic Guidance for Surface Deformation", *Computer Graphics Forum*, Vol. 24, No. 3, pp. 601--609.

[10] Masuda, H., Yoshioka, Y. and Furukawa, Y., 2006, "Interactive Mesh Deformation Using Equality-Constrained Least Squares", *Computers and Graphics*, Vol. 30, No. 6, pp. 936-946.

[11] Masuda, H., Yoshioka, Y. and Furukawa, Y., 2006, "Preserving Form-Features in Interactive Mesh Deformation", *LNCS 4077 (Proceedings of Geometric Modeling and Processing - GMP2006)*, pp. 207-220.

[12] Gould, N.I.M., Hu, Y. and Scott, J.A., 2005, "A Numerical Evaluation of Sparse Direct Solvers for the Solution of Large Sparse, Symmetric Linear Systems of Equations", *Technical Report RAL-TR-2005-005, Rutherford Appleton Laboratory.*

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

[14] Demmel, J., Gilbert, J. and Li, X., 1995, "SuperLU User's Guide".

[15] Botsch, M., Bommes, D. and Kobbelt, L., 2005, "Efficient Linear System Solvers for Mesh Processing", *IMA Conference on the Mathematics of Surfaces 2005*, pp. 62–83.

[16] Cavendish, J. C., 1995, "Integrating feature-based surface design with freeform deformation", *Computer-Aided Design,* 703-711.

[17] Fontana, M., F.,Meirana, M. , 1999, "A free-form feature taxonomy", *The Computer Graphics Forum,* pp.107–118.

[18] Pernot, J.P., Guillet, S., Léon, J.C., Catalano, C.E., Giannini, F. and Falcidieno, B., 2002, "A Shape Deformation Tool to Model Character Lines in the Early Design Phases", *The Shape Modeling International Conference*, 165–173.

[19] Meyer, M., Desbrun, M., Schröder, P. and Barr, A. H., 2003, "Discrete Differential-Geometry Operators for Triangulated 2-Manifolds", *Proceedings of Visualization and Mathematics III*, pp. 35–57.

[20] Pinkall, U. and Polthier, K., 1993, "Computing Discrete Minimal Surfaces and Their Conjugates", *Experimental Mathematics*, Vol. 2, No. 1, pp. 15–36.

[21] Alexa, M., 2003, "Differential Coordinates for Local Mesh Morphing and Deformation", *The Visual Computer*, Vol. 19, No. 2-3, pp.105–114.