Preserving Form Features in Interactive Mesh Deformation

Hiroshi Masuda [a,*], Yasuhiro Yoshioka [a], Yoshiyuki Furukawa [b]

[a]*The University of Tokyo, Graduate School of Engineering, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan*

[b]*National Institute of Advanced Industrial Science and Technology, Digital Manufacturing Research Center, Namiki, Tsukuba-shi, Ibaraki 305-8561, Japan*

Abstract

Interactive mesh deformation that preserves differential properties is a promising technique for the design of mechanical parts such as automobile sheet-metal panels. However, existing methods lack the ability to manipulate the form features and hard constraints that are commonly used in engineering applications. In this paper, we propose a new deformation framework that precisely preserves the shapes of form features during deformation. Geometric shapes are interactively deformed so that mean curvature normals are approximately preserved in a least-squares sense and positional constraints and form-feature constraints are precisely satisfied. In our system, the combination of soft and hard constraints is solved using the Lagrange multiplier method. We also show how to constrain the motion of a form feature on a plane or a straight line using linear constraints. The implemented system achieves a real-time response for constrained deformation.

Key words: interactive mesh deformation, form-feature, linear constraints, Laplacian coordinates, mean curvature normal

1 Introduction

In product design, 3D models are often created in the early stage of product development and used for design evaluation by the development team. The evaluation of design concepts in the early stage helps products to meet manufacturing, cost,

* Corresponding author.
Email addresses: masuda@nakl.t.u-tokyo.ac.jp (Hiroshi Masuda), yoshioka@nakl.t.u-tokyo.ac.jp (Yasuhiro Yoshioka), y-furukawa@aist.go.jp (Yoshiyuki Furukawa).

safety, quality and maintenance requirements. Since design concepts are very often changed or discarded in the early stage, it is not reasonable to spend a great deal of time on creating detailed 3D models. Interactive and intuitive 3D modeling tools are preferable in the early design stage. Parametric surfaces, such as Bézier, B-splines and NURBS, have been widely used to represent free-form shapes in CAD applications, but it is very tedious and time-consuming to manipulate surface patches with a large number of control points.

Our motivation for studying interactive deformation stems from requirements for the deformation of automobile sheet-metal panels. Many interactive shape deformation techniques have been developed so far, but they do not necessarily meet engineering requirements. In sheet metal panels, regions such as circle holes and character lines are often required to retain their original shapes for manufacturing, assembly or aesthetic reasons. We call such partial regions *form features*. Sheet metal panels typically consist of a combination of smooth base surfaces and form features, each of which has different characteristics; while smooth base surfaces are characterized by a curvature distribution, form features are defined by surface types and their parameters [1,2]. Therefore, in engineering design it is necessary to control the curvature of base surfaces and the shapes of form features during shape deformation.

Volume-based deformation is a popular interactive technique used in computer graphics applications [3–5]. Such techniques change geometric shapes by deforming the space in which they lie. However, it is not easy for the user to generate intended geometric shapes by manipulating the control lattices of volumes.

In the last few years, several surface-based deformation techniques have been proposed [6–8]. These methods calculate differential properties on a surface and encode the geometric shape using partial differential equations. In a typical deformation method, the user first selects the fixed region, which remains unchanged, and the handle region, which is used as the manipulation handle, and drags the handle region on the screen. Then all vertex positions in the mesh model are calculated according to the position and orientation of the handle region. During deformation, the system interactively solves partial differential equations by treating the fixed and handle regions as boundary conditions.

Surface-based deformation techniques are useful for deforming free-form surfaces, but existing methods cannot preserve the shapes of form features. For example, circular screw holes may be deformed to ellipses. In addition, existing methods solve all constraints using the least-squares method and therefore produce compromise solutions. However, many engineering applications require precise satisfaction of dimensional and shape constraints. It is possible to set large weights for certain constraints in a least-squares system [9], but very large weights may cause numerical problems. It is difficult to predict adequate weight values that satisfy the allowable margin of error.

Instead, we introduce soft and hard constraints in shape deformation. When constraints are approximately satisfied in a least-squares sense, we call them *soft constraints*, and when constraints are precisely satisfied, we call them *hard constraints*. In typical shape design, while differential properties are not explicitly specified by the user, form features and positional constraints are user-specified. It is obviously reasonable to treat differential properties as soft constraints and user-defined constraints as hard constraints.

In this paper, we propose a novel mesh-editing framework that can manage form features precisely. Our main contributions in this paper are:

- A novel deformation framework in which hard and soft constraints are incorporated and rotations are propagated using quaternion logarithms;
- The introduction of new constraints for translating and rotating form features while preserving their original shapes; and
- The introduction of new constraints for maintaining the motion of form features on a straight line or a plane.

In the following section, we review the related work on mesh deformation. In Section 3, we describe our mesh deformation framework, in which hard constraints are incorporated using Lagrange multipliers. Then a new rotation method is introduced based on quaternion logarithms. In Section 4, we introduce a method for preserving form features by constraining the relative positions and rotations of vertices, and then propose new feature constraints that maintain the motion of a form feature on a line or a plane. We also present a simple feature extraction method. In Section 5, we evaluate our framework and show experimental results. We conclude the paper in Section 6.

2 Related Work

Interactive mesh deformation techniques have been intensively studied. Such research aims to develop modeling tools that intuitively modify free-form surfaces while preserving the details of shapes. There are several types of approach, which are based on space deformation (FFD), multiresolution, and partial differential equations (PDE).

FFD is very popular in computer graphics. Such methods modify shapes by deforming the 3D space in which objects lie [3–5]. Cavendish [2] discussed FFD approaches in the context of design support and reported that FFD could be used for designing automotive sheet-metal panels. Our aim is also to support the design of automotive sheet-metal panels, but in our empirical investigation of the automobile industry, problems arise when shapes are deformed using FFD because form-features in a 3D model are deformed in unintended ways. It is difficult for

FFD approaches to manage constraints on form features because the manipulation handles do not work directly on geometric shapes.

Multiresolution approaches [10–14] decompose a surface into a base mesh and several levels of detail, each of which is represented as the difference between successive resolution levels. A shape is globally deformed at low resolution and locally deformed at high resolution. Botsch and Kobbelt [7] applied this technique to interactive mesh editing. A mesh model is decomposed into two-level resolutions and the smooth base is interactively deformed using energy minimization techniques. Geometric details are then recovered on the modified smooth shape. However, it is difficult to control the shapes of form features precisely in a multiresolution framework.

PDE-based approaches directly deform the original mesh based on geometric constraints. These methods are categorized as non-linear and linear methods.

Non-linear methods typically solve Laplacian or Poisson equations using non-linear iterative solvers [15–19]. Catalano et al. [20] investigated support tools for aesthetic design and found that non-linear PDE approaches are popular in computer-aided design. These methods produce fair surfaces, but they are time-consuming and difficult to use in an interactive environment.

Linear PDE-based approaches represent differential properties and positional constraints as a linear system. Discrete Laplacian operators are often used to represent differential properties [6,7]. Yu et al. [8] introduced a similar technique called Poisson editing, which manipulates the gradients of the coordinate functions of the mesh. Zhou et al. [21] proposed volumetric Laplacian operators for large deformations. We believe that linear PDE-based approaches are useful for product design when product shapes undergo frequent design modifications.

A discrete Laplacian operator on a mesh is defined as the difference vector between a vertex position and the weighted average positions of its one-ring neighbors. Since Laplacians are defined in the local coordinate systems [6,22], one or more vertices must be specified in the global coordinate system to determine all vertex positions. Since the number of differential equations and positional constraints is larger than the number of vertices, the least-squares method is typically used to calculate compromise solutions. When Laplacians and positional constraints are described as a linear system $M\mathbf{x} = \mathbf{b}$, the least-squares system is represented as $M^t M\mathbf{x} = M^t\mathbf{b}$, and therefore vertex positions are calculated by solving this linear system. Since $M^t M$ is a sparse symmetrical positive definite matrix, it can be efficiently factorized [23]. After the matrix is factorized once, $\mathbf{x}$ is interactively calculated according to the modification of $\mathbf{b}$.

The least-squares method is effective, but is not useful when some design parameters must be precisely satisfied. Welch and Witkin [24] introduced a combination of soft and hard constraints in variational surface modeling and solved them using Lagrange multipliers, but they did not apply these to interactive mesh editing. Yoshioka et al. [25] solved hard constraints using equality-constrained least squares. This method is very effective when constraints are restricted to simple positional constraints. However, it is not effective for form-feature constraints, because a large number of hard constraints with two or more variables make equality-constrained least-squares systems less sparse. In this paper, we introduce hard constraints and new form-feature constraints for interactive mesh editing and solve them using the Lagrange multiplier.

Several authors have discussed methods for rotating Laplacian vectors following the deformation of surfaces. Since the rotation is applied to **b** on the right-hand side of the least-squares system, Laplacian vectors can be rotated interactively. Lipman et al. [26] estimated the local rotations on the underlying smooth surface. Sorkine et al. [6] linearized elements in a rotation matrix assuming that the rotation angles are very small. Lipman et al. [27] defined a local frame for each vertex and encoded rotations using the relative orientations of these local frames. Zayer et al. [28] rotated Laplacian vectors using harmonic functions in $[0,1]$ based on discrete Laplace–Beltrami operators. They defined a unit quaternion at each vertex and interpolated four components of unit quaternions by assigning a single weight 1.0 to all handle vertices. Our approach is similar to the Zayer method, although we assign the logarithms of unit quaternions to vertices.

3 Framework for Constrained Deformation

3.1 Constraints on positions

Let mesh $\mathbb{M}$ be a pair $(K,\mathbb{P})$, where $\mathbb{P} = \{\mathbf{p}_1,\ldots,\mathbf{p}_n\}$ and $\mathbf{p}_i = (x_i,y_i,z_i) \in \mathbb{R}^3$; K is a simplicial complex that contains vertices i, edges (i,j), and faces (i,j,k). The adjacent vertices of vertex i are denoted by $N(i) = \{j|(i,j) \in K\}$. The original position of $\mathbf{p}_i$ is referred to as $\mathbf{p_i^0} = (x_i^0,y_i^0,z_i^0)$.

When the normal vector and mean curvature of vertex i are referred to as κ_i and $\mathbf{n}_i$, the mean curvature normal $\kappa_i\mathbf{n}_i$ can be approximated using the following discrete form [29]:

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

where α_{ij} and β_{ij} are the two angles opposite to the edge in the two triangles that share edge (i,j), as shown in Figure 1. We denote the mean curvature normal of vertex i in the original mesh as δ_i.

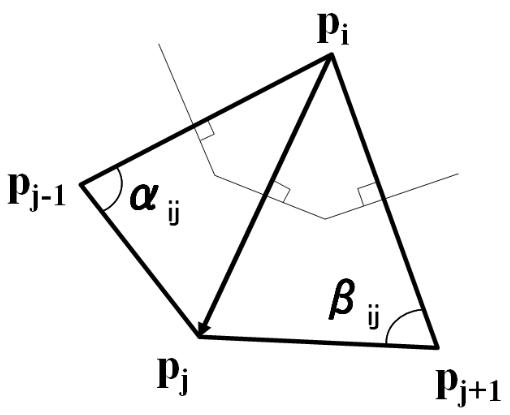

Fig. 1. Definition of α_{ij} and β_{ij}.

In this paper, we describe user-defined constraints other than mean curvature normals as $\mathbf{f}_j(\mathbb{P}_j) = \mathbf{u}_j \; (\mathbb{P}_j \subset \mathbb{P})$. Then constraints for vertices can be described using the following linear equations:

$$\begin{cases} L(\mathbf{p}_i) = R(\mathbf{n}_i, \theta_i)\delta_i \quad (i = 1, 2, \ldots, n) \\ \mathbf{f}_j(\mathbb{P}_j) = R(\mathbf{m}_j, \phi_j)\mathbf{u}_j \quad (j = 1, 2, \ldots, m), \end{cases} \tag{2}$$

where n is the number of vertices, m is the number of user-defined constraints, and $R(\mathbf{n}, \theta)$ represents a rotation matrix that rotates a vector around axis $\mathbf{n} \in \mathbb{R}^3$ by angle $\theta \in \mathbb{R}$. $R(\mathbf{n}_i, \theta_i)$ and $R(\mathbf{m}_j, \phi_j)$ are calculated before Equation (2) is solved. We describe a method for calculating rotation matrices in the next section.

Since the number of constraints in Equation (2) is greater than the number of variables, no exact solution exists. Therefore, we classify constraints in Equation (2) into soft and hard constraints.

When we represent soft constraints as $A\mathbf{x} = \mathbf{b}$ and hard constraints as $C\mathbf{x} = \mathbf{d}$ in matrix form, variables $\mathbf{x}$ that minimize $||A\mathbf{x} - \mathbf{b}||^2$ subject to $C\mathbf{x} = \mathbf{d}$ can be calculated using the Lagrange multiplier as:

$$\min_{\mathbf{x}} \left(\frac{1}{2}||A\mathbf{x} - \mathbf{b}||^2 + \mathbf{y}^t(C\mathbf{x} - \mathbf{d})\right), \tag{3}$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_{3m})^t$ are Lagrange multipliers. This minimization can be calculated using the following linear system:

$$M\tilde{\mathbf{x}} = \tilde{\mathbf{b}} \tag{4}$$

$$M = \begin{pmatrix} A^tA & C^t \\ C & 0 \end{pmatrix}, \tilde{\mathbf{x}} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}, \tilde{\mathbf{b}} = \begin{pmatrix} A^t\mathbf{b} \\ \mathbf{d} \\ . \end{pmatrix}$$

This linear system determines the unique solution that precisely satisfies the hard constraints. Matrix M can be factorized using sparse direct solvers for linear symmetric systems [30].

We note that when conflicting or redundant constraints are involved in the linear system of hard constraints, they lead to rank deficiency of the linear system and the solver may halt the computation. Such over-constraint problems can be resolved by applying Householder factorization to each column in matrix C, as shown by Yoshioka et al. [25]. If the jth column in C^t is a redundant constraint, the diagonal and lower elements of the jth column are equal to zero after the previous $j-1$ columns are processed. Therefore, we can detect the redundant constraints. This process can be calculated very efficiently (see Ref. [25] for more details).

3.2 Constraints on rotations

In this section, we describe our new rotation-propagation method for calculating $R(\mathbf{n}_i, \theta_i)$ in Equation (2).

In our framework, we assign the logarithms of unit quaternions to all vertices. A quaternion can be written in the form:

$$Q = (w, x, y, z) = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}, \tag{5}$$

where $w, x, y, z \in \mathbb{R}$ and $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are distinct imaginary numbers. When a quaternion has unit magnitude, it is called a unit quaternion and corresponds to a unique rotation matrix. A unit quaternion can be represented using rotation axis $\hat{\mathbf{n}}$ and rotation angle θ as:

$$\hat{Q} = cos\frac{\theta}{2} + \hat{\mathbf{n}}sin\frac{\theta}{2} = e^{\hat{\mathbf{n}}\frac{\theta}{2}}, \tag{6}$$

where $\hat{\mathbf{n}}$ is a pure quaternion. The logarithm of a unit quaternion is defined as the inverse of the exponential:

$$\mathbf{q} = ln\hat{Q} = \frac{\theta}{2}\hat{\mathbf{n}}. \tag{7}$$

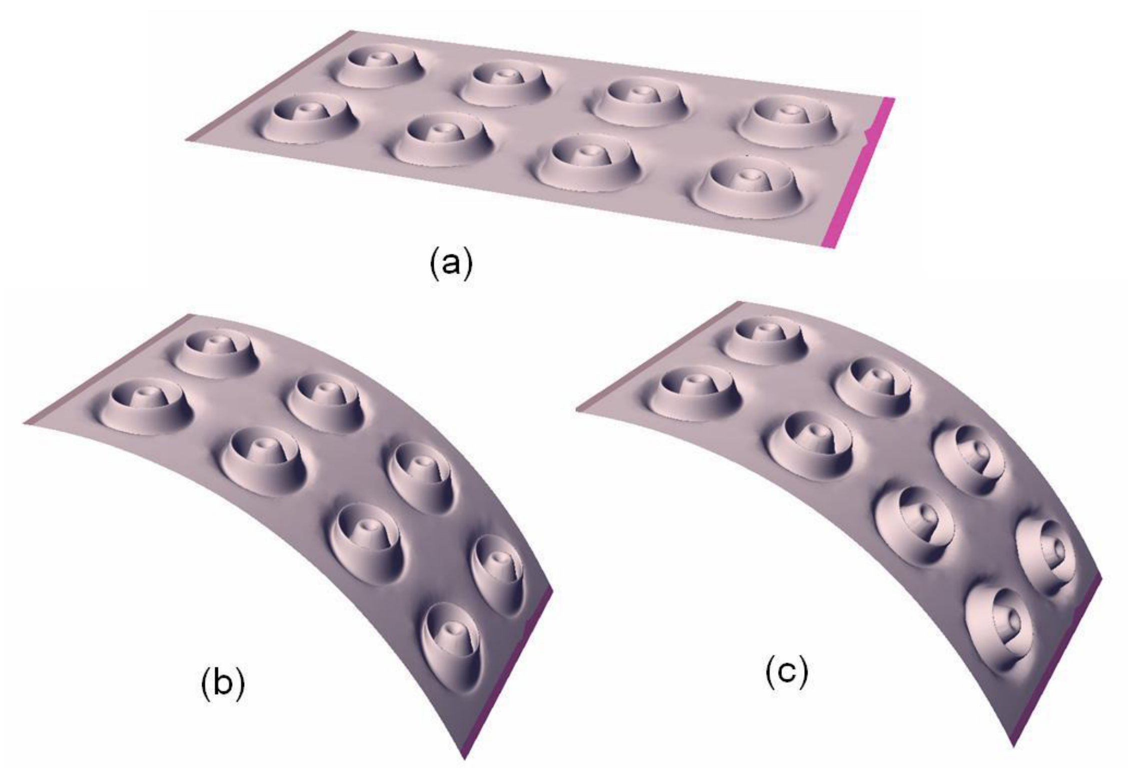

Fig. 2. Deformed shapes. (a) Original shape; (b) deformed shape without the rotation of normals; and (c) deformed shape with the rotation of normals.

We assign logarithm $\mathbf{q}_i \in \mathbb{R}^3$ to vertex i and denote the logarithms assigned to all vertices as $\mathbb{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n\}$. When $\mathbf{q}_i$ is equal to 0, the mean curvature normal is not rotated; when $\mathbf{q}_i = \mathbf{v}_j$ is specified, the mean curvature normal is rotated around axis $\mathbf{v}_j/|\mathbf{v}_j|$ by angle $2|\mathbf{v}_j|$.

Shoemake [31] proposed spherical linear interpolation between two unit quaternions. Johnson [32] applied spherical linear interpolation to multiple unit quaternions using logarithms of the unit quaternions. Pinkall and Polthier [33] proposed an interpolation technique using discrete conformal mapping. Zayer et al. [28] applied discrete conformal mapping to the interpolation of unit quaternions. We introduce similar constraints on the logarithms of unit quaternions:

$$L(\mathbf{q}_i) = \frac{1}{4A_i} \sum_{j \in N(i)} (cot\,\alpha_{ij} + cot\,\beta_{ij})(\mathbf{q}_i - \mathbf{q}_j) = 0. \tag{8}$$

Then we assign $\mathbf{q}_i = 0$ at each fixed vertex and $\mathbf{q}_i = \mathbf{v}_i$ at each handle vertex, where $\mathbf{v}_j$ is a quaternion logarithm specified by the user. We generally describe these constraints as:

$$\mathbf{g}_j(\mathbb{Q}_j) = \mathbf{v}_j \quad (\mathbb{Q}_j \subset \mathbb{Q}, \mathbf{v}_j \in \mathbb{R}^3). \tag{9}$$

As a result, linear equations for rotations can be described as:

$$\begin{cases} L(\mathbf{q}_i) = 0 \quad (i = 1, 2, \ldots, n) \\ \mathbf{g}_j(\mathbb{Q}_j) = \mathbf{v}_j \quad (j = 1, 2, \ldots, m), \end{cases} \tag{10}$$

where n is the number of vertices and m is the number of user-defined constraints on rotations. These equations construct a sparse linear system and can be solved using sparse direct solvers [30]. The solution of this linear system generates certain energy-minimization surfaces [33] in 3D space spanned by logarithms of the unit quaternions.

When all components of $\mathbb{Q}$ are calculated, a rotation matrix $R(\mathbf{q}_i/|\mathbf{q}_i|, 2|\mathbf{q}_i|)$ is uniquely determined for each vertex. Figure 2 shows a mesh model that contains patterns on a plane. While the patterns with constant normals are distorted, as shown in Figure 2b, the ones in Figure 2c are smoothly deformed because the mean curvature normals are rotated according to rotation of a handle region.

4 Preserving Form Features

4.1 Preserving the shapes of form features

We define a form feature as a partial shape that has an engineering meaning, such as a hole or a protrusion. In the mesh model $\mathbb{M}(K, \mathbb{P})$, a form feature is a submesh that consists of the simplicial subcomplex of K and the subset of vertices $\mathbb{P}$. We denote a form feature as F and the index set of vertices in form feature F as Λ_F. We obtain a spanning tree by traversing edges in a form-feature region, as shown in Figure 3. Spanning trees are used to avoid redundant constraints for form features. We denote edges in the spanning tree as T_F.

When form feature F is translated and rotated while preserving the original shape, the following constrains need to be added for the rotations and vertex positions:

$$\begin{cases} \mathbf{q}_i - \mathbf{q}_j = 0 \quad (i, j \in \Lambda_F; (i, j) \in T_F) \\ \mathbf{p}_i - \mathbf{p}_j = s_F R(\mathbf{n}_i, \theta_i)(\mathbf{p}_i^0 - \mathbf{p}_j^0) \quad (i, j \in \Lambda_F; (i, j) \in T_F), \end{cases} \tag{11}$$

where s_F is the scaling factor of the form feature. The first equation in (11) shows that $\mathbf{q}_i$ must be the same in the form-feature region, because each vertex in the form feature has the same rotation matrix. The second equation in (11) preserves the relative positions of the vertices in the form feature. These equations for rotations and positions are added to Equations (2) and (10).

In some cases, a form feature has to retain the original direction, depending on the design intention. Then the following equations are added to (2) and (10) instead of Equation (11):

$$\begin{cases} \mathbf{q}_i = 0 \quad (i \in \Lambda_F) \\ \mathbf{p}_i - \mathbf{p}_j = s_F(\mathbf{p}_i^0 - \mathbf{p}_j^0) \quad (i, j \in \Lambda_F; (i, j) \in K). \end{cases} \tag{12}$$

Figure 4 shows deformed shapes that contain circle holes. The fixed and handle regions are shown in Figure 4a. While the unconstrained circles in Figure 4b are stretched, the circle shapes in Figure 4c–e are maintained by constraints on relative positions. In Figure 4e, four small holes are constrained by Equation (11), but the center hole is constrained to preserve the original direction using Equation (12).

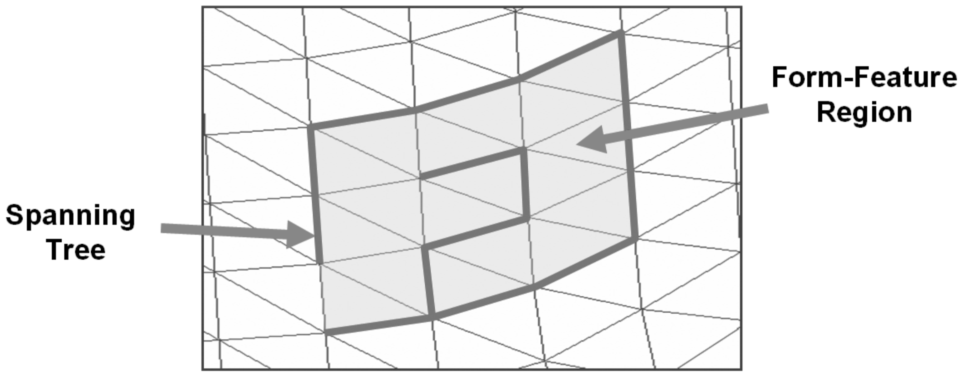

Fig. 3. Spanning tree in a form-feature region.

4.2 Constraining the motion of form features

In computer-aided design, it is useful to constrain the motion of a form feature. The positions of form features are often specified using datum lines or planes.

The motion of a form feature can be maintained on a straight line or a plane by constraining a point in the form feature. A constrained point can be specified as a linear function of vertex positions. For example, the center position of a circle can be specified as the function $0.5(\mathbf{p}_i + \mathbf{p}_j)$ using two vertex positions on opposite sides. Here, we simply represent a linear combination of coordinates $\{\mathbf{p}_i\}(i \in \Lambda_F)$ as $\mathbf{x} = (x, y, z)$.

4.2.1 On-plane constraints

A plane is uniquely determined by its normal vector and a point on the plane. We represent the equation of a plane as $\mathbf{n}(\mathbf{x} - \mathbf{p}) = 0$, where $\mathbf{n} = (n_x, n_y, n_z)$ is

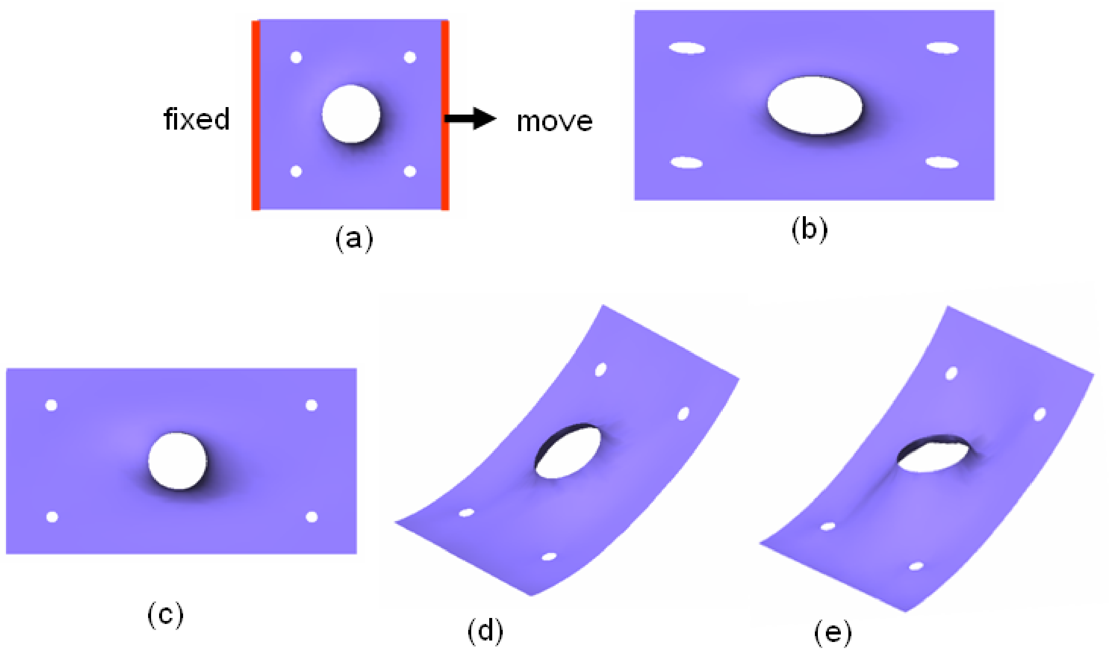

Fig. 4. Constrained deformation. (a) Original shape; (b) deformed shape with no form-feature constraints; (c) stretched shape with the shapes of circles preserved; (d) deformed shape with five rotated circles; and (e) deformed shape with the direction of the center circle preserved.

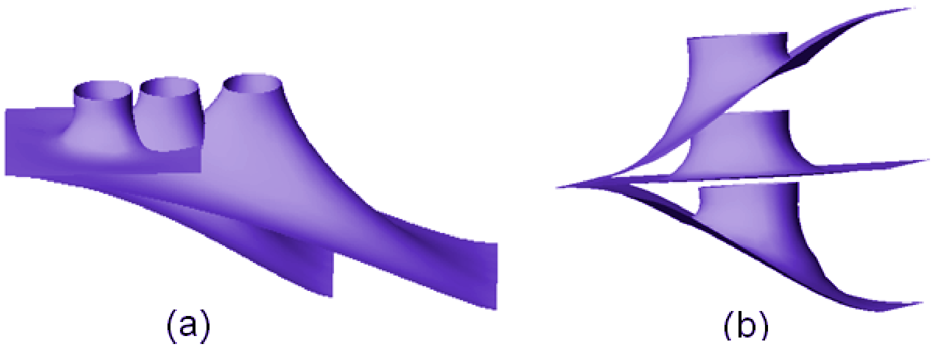

Fig. 5. Form feature moved on (a) a plane and (b) a straight line.

the normal vector and $\mathbf{p} = (p_x, p_y, p_z)$ is a point on the plane. Then the following equation constrains the motion of a form feature on the plane:

$$n_x x + n_y y + n_z z = n_x p_x + n_y p_y + n_z p_z. \tag{13}$$

Since position $\mathbf{p}$ appears on the right-hand side of Equation (13), planes can be interactively moved to their normal directions.

Figure 5a shows a deformed shape in which a hole feature is constrained on a plane when the handle region is moved.

4.2.2 On-line constraints

A straight line can be represented as $\mathbf{x} = k\mathbf{n} + \mathbf{p}$. Let $\mathbf{l}$, $\mathbf{m}$ and $\mathbf{n}$ be unit vectors that are perpendicular to each other. Then position (x, y, z) moves on the straight line using the following constraints:

$$\begin{cases} l_x x + l_y y + l_z z = l_x p_x + l_y p_y + l_z p_z \\ m_x x + m_y y + m_z z = m_x p_x + m_y p_y + m_z p_z. \end{cases} \tag{14}$$

Straight lines can be moved interactively to directions that are perpendicular to $\mathbf{n}$. This capability is useful for moving the centers of circles on 2D drawings.

In Figure 5b, a form feature moves on a straight line according to the motion of the handle region.

4.3 Feature extraction

In our system, the user selects the form-feature regions and then adds linear constraints to the form features. It may be tedious work for the user to carefully select the region before specifying constraints. Therefore, we introduce an interactive mesh segmentation technique for easy selection of form-feature regions.

So far, many segmentation algorithms have been reported [34–38]. Our segmentation algorithm is based on the method proposed by Katz and Tal [37], but we apply the method only to user-specified regions [39].

Figure 6 shows a feature extraction process. First, the user roughly selects a region that includes the boundary of a feature region, as shown in Figure 6b. The region must be selected so that the mesh model is exactly separated into two regions. Then the optimal cut is calculated by the maximum-flow, minimum-cut algorithm [37]. Finally, the feature region is separated, as shown in Figure 6c.

5 Experimental Results

Figure 7 shows examples of some deformed shapes. In industrial design, character lines are extremely important. If a deformation process modifies the character lines of a product shape, the resultant shape will not be accepted by the designers. In

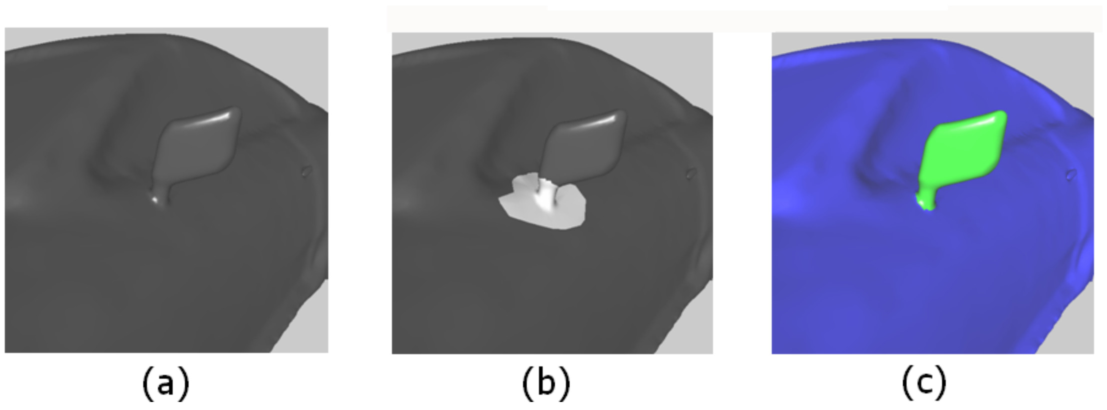

Fig. 6. Feature extraction: (a) original shape; (b) region selected by the user; and (c) the extracted region.

Figure 7a, no form features are specified and the character lines are warped in unintended ways. In Figure 7b, the character lines are specified as form features and are thus preserved after the shape is deformed.

Figure 8 shows the front grille part of an automobile model. While Figure 8b contains no form-feature constraints, Figure 8c has form-feature constraints around the cavities. Deformation in Figure 8b destroys the design intent, but Figure 8c maintains the shapes of the cavities. In Figure 8d, the scaling factors of form features in Equation 12 are modified in an interactive manner.

Figure 9 shows a sheet metal panel. The 16 cavities shown by arrows are constrained so that they rotate while the original shapes are preserved.

Table 1 shows the CPU time for calculating the deformed models in Figures 7–9. Factorization of matrices was performed using SuperLU [40], which is a sparse linear system solver based on LU decomposition. The CPU time was measured for setting up matrices and factorizing them on a 1.50-GHz Pentium-M PC with 1 GB of RAM. Once the matrix was set up and factorized, the shape could be deformed at an interactive rate. This result shows that the performance of our framework is adequate for interactive applications.

	Vert	Soft	Hard	Feat	Time
Figure 7	3337	5796	3826	3702	0.86
Figure 8	13974	25458	9334	1072	4.69
Figure 9	2982	6084	3308	3202	0.99

Table 1
CPU time for setting up and factorizing linear systems. Vert: number of vertices; Soft: number of soft constraints; Hard: number of hard constraints; Feat: number of form-feature constraints; Time: CPU time (s).

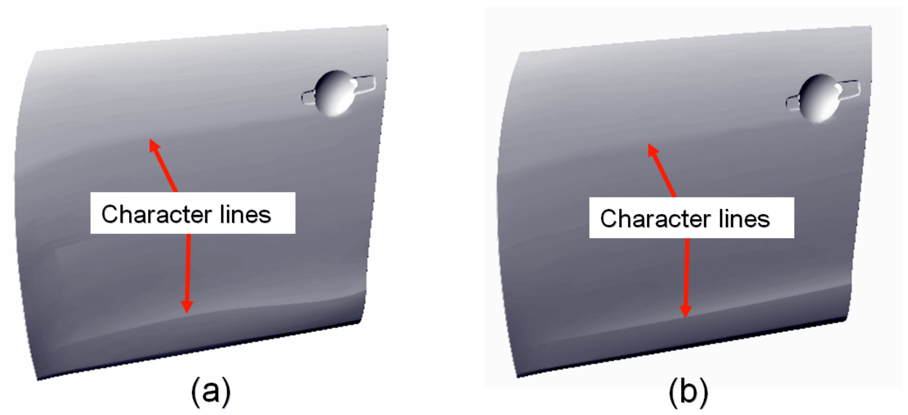

Fig. 7. Door panel. (a) Shape deformed without form-feature constraints. The character lines are warped. (b) Shape deformed with form-feature constraints on the character lines.

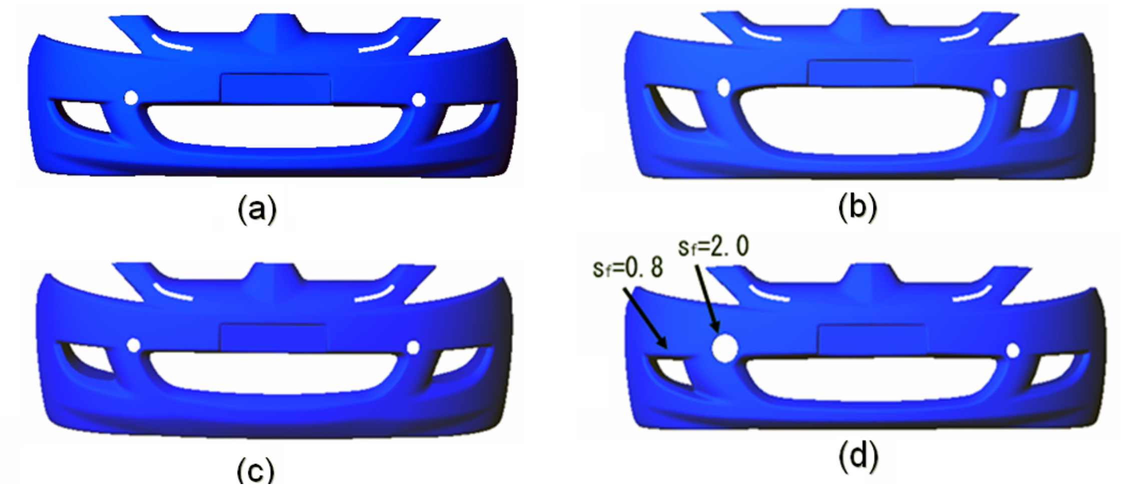

Fig. 8. Front grille: (a) original shape; (b) shape deformed without form-feature constraints; (c) shape deformed with form-feature constraints; and (d) interactive scaling of form features.

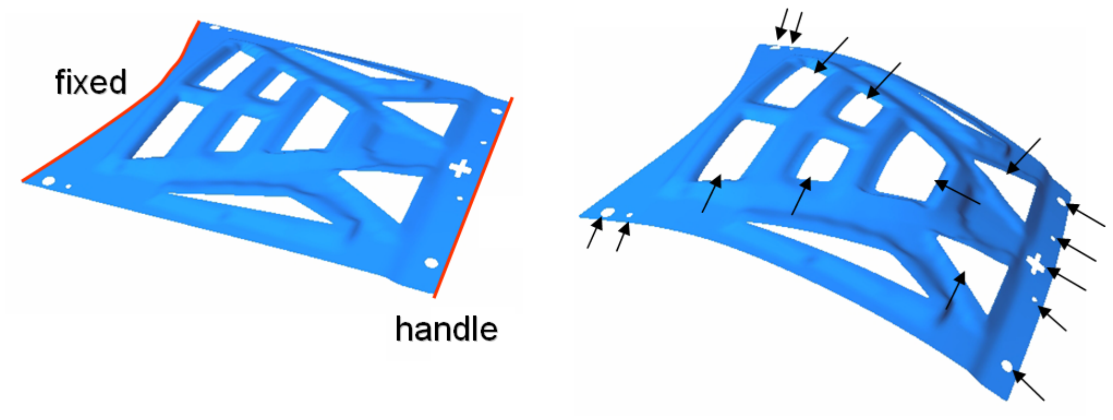

Fig. 9. Sheet metal part. Top: original planar shape. Bottom: deformed shape with rotated form features shown by arrows.

6 Conclusions and Future Work

We have presented a discrete framework for incorporating constraints of form features using a hard constraints approach. We solved a combination of soft and hard constraints using the Lagrange multiplier method. We rotated mean curvature normals and form features by interpolating quaternion logarithms. Constraints on rotations and positions are separately solved as two sparse symmetrical matrices, which are known for the existence of efficient solvers. We showed how to constrain the shape and motion of form features; shape constraints can preserve the shapes of form features and motion constraints confine movement on a plane or a straight line. These constraints are convenient for deforming 3D models while preserving form features.

In future work, it will be important to develop more intuitive GUI tools such as sketch-based interfaces [41]. In addition, it will be useful to incorporate a geometric reasoning engine into our framework for modifying the shapes of form features using parameters. Our current solver is based on SuperLU, which is relatively slow for very large systems [30]. More efficient linear solvers may be required to deform very large models. Finally, our framework handles only linear constraints, while some design constraints require non-linear equations. We would like to investigate how to handle non-linear constraints in an interactive environment.

Acknowledgements

This work was partly funded by Mitsubishi Motors Corporation (MMC) and the 3D models of automobile parts in Figures 7 and 8 are reproduced courtesy of MMC.

References

[1] Fontana, M., Giannini, F., Meirana, M. A free-form feature taxonomy. Computer Graphics Forum 1999;18(3):107–118.

[2] Cavendish, J. C. Integrating feature-based surface design with freeform deformation. Computer-Aided Design 1995;27(9):703–711.

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

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

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

[6] Sorkine, O., Lipman, Y., Cohen-Or, D., Alexa, M., Rössl, C., Seidel, H.-P. Laplacian surface editing. in: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing. 2004. pp. 175–184.

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

[8] 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 2004;23(3):644–651.

[9] Sorkine, O. Laplacian mesh processing. in: STAR Proceedings of Eurographics 2005. 2005. pp. 53–70.

[10] Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., Stuetzle, W. Multiresolution analysis of arbitrary meshes. in: Proceedings of SIGGRAPH 1995. 1995. pp. 173–182.

[11] Zorin, Z., Schröder, P., Sweldens, W. Interactive multiresolution mesh editing. in: Proceedings of SIGGRAPH 1997. 1997. pp. 259–268.

[12] Kobbelt, L., Campagna, S., Vorsatz, J., Seidel, H.-P. Interactive multi-resolution modeling on arbitrary meshes. in: Proceedings of SIGGRAPH 1998. 1998. pp. 105–114.

[13] Guskov, I., Sweldens, W., Schröder, P. Multiresolution signal processing for meshes. in: Proceedings of SIGGRAPH 1999. 1999. pp. 325–334.

[14] Lee, S. Interactive multiresolution editing of arbitrary meshes. Computer Graphics Forum 1999;18(3):73–82.

[15] Bloor, M. I. G., Wilson, M. J. Using partial differential equations to generate free-form surfaces. Computer-Aided Design 1990;22(4):202–212.

[16] Schneider, R., Kobbelt, L. Generating fair meshes with g1 boundary conditions. in: Proceedings of the 2000 International Conference on Geometric Modeling and Processing. 2000. pp. 251–261.

[17] Desbrun, M., Meyer, M., Schröder, P., Barr, A. H. Implicit fairing of irregular meshes using diffusion and curvature flow. in: Proceedings of SIGGRAPH 1999. 1999. pp. 317–324.

[18] Yamada, A., Furuhata, T., Shimada, K., Hou, K.-H. A discrete spring model for generating fair curves and surfaces. in: Pacific Conference on Computer Graphics and Applications. 1999. pp. 270–279.

[19] Taubin, G. A signal processing approach to fair surface design, in: Proceedings of SIGGRAPH 1995. 1995. pp. 351–358.

[20] Catalano, C. E., Falcidieno, B., Giannini, F., Monti, M. A survey of computer-aided modeling tools for aesthetic design. Journal of Computer and Information Science in Engineering 2002;2(11):11–20.

[21] 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 2005;24(3):496–503.

[22] Alexa, M. Differential coordinates for local mesh morphing and deformation. The Visual Computer 2003;19(2–3):105–114.

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

[24] Welch, W., Witkin, A. Variational surface modeling. in: Proceedings of SIGGRAPH 1992. 1992. pp. 157–166.

[25] Yoshioka, Y., Masuda, H., Furukawa, Y. A constrained least-squares approach to interactive mesh deformation. in: Proceedings of the 2006 International Conference on Shape Modeling and Applications. 2006. pp. 153–162.

[26] Lipman, Y., Sorkine, O., Cohen-Or, D., Levin, D., Rössl, C., Seidel, H.-P. Differential coordinates for interactive mesh editing. in: Proceedings of the 2004 International Conference on Shape Modeling and Applications. 2004. pp. 181–190.

[27] Lipman, Y., Sorkine, O., Levin, D., Cohen-Or, D. Linear rotation-invariant coordinates for meshes. ACM Transactions on Graphics 2005;24(3):479–487.

[28] Zayer, R., Rössl, C., Karni, Z., Seidel, H.-P. Harmonic guidance for surface deformation. Computer Graphics Forum 2005;24(3):601–609.

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

[30] Gould, N. I. M., Hu, Y., Scott, J. A. A numerical evaluation of sparse direct solvers for the solution of large sparse, symmetric linear systems of equations. Technical Report RAL-TR-2005-005. Council for the Central Laboratory of the Research Councils. 2005.

[31] Shoemake, K. Animating rotation with quaternion curves. in: Proceedings of SIGGRAPH 1985. 1985. pp. 245–254.

[32] Johnson, M. P. Exploiting quaternions to support expressive interactive character motion. Ph.D. thesis. Massachusetts Institute of Technology, School of Architecture and Planning, Program in Media Arts and Sciences. 2003.

[33] Pinkall, U., Polthier, K. Computing discrete minimal surfaces and their conjugates. Experimental Mathematics 1993;2(1):15–36.

[34] Mangan, A. P., Whitaker, R. T. Partitioning 3D surface meshes using watershed segmentation. IEEE Transaction on Visualization and Computer Graphics 1999;5(4):308–321.

[35] Chazelle, B., Dobkin, D. P., Shouraboura, N., Tal, A. Strategies for polyhedral surface decomposition: An experimental study. Computational Geometry: Theory and Applications 1997;7(4–5):327–342.

[36] Shlafman, S., Tal, A., Katz, S. Metamorphosis of polyhedral surfaces using decomposition. Computer Graphics Forum 2002;21(3):219–228.

[37] Katz, S., Tal, A. Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Transactions on Graphics 2003;22(3):954–961.

[38] Katz, S., Leifman, G., Tal, A. Mesh segmentation using feature point and core extraction. The Visual Computer 2005;21(8–10):649–658.

[39] Masuda, H., Furukawa, Y., Yoshioka, Y., Yamato, H. Volume-based cut-and-paste editing for early design phases. in: ASME 2004 Design Engineering Technical Conference and Computer and Information Engineering Conference. 2004.

[40] Demmel, J., Gilbert, J., Li, X. SuperLU User's Guide. 1995.

[41] Nealen, A., Sorkine, O., Alexa, M., Cohen-Or, D. A sketch-based interface for detail-preserving mesh editing. ACM Transactions on Graphics 2005;24(3):1142–1147.