Interactive Mesh Deformation Using Equality-Constrained Least Squares

H. Masuda [a,*], Y. Yoshioka [a], Y. Furukawa [b]

[a] *The University of Tokyo, Department of Environmental and Ocean 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

Mesh deformation techniques that preserve the differential properties have been intensively studied. In this paper, we propose an equality-constrained least squares approach for stably deforming mesh models while approximately preserving mean curvature normals and strictly satisfying other constraints such as positional constraints. We solve the combination of hard and soft constraints by constructing a typical least squares systems using QR decomposition. A well-known problem of hard constraints is over-constraints. We show that the equality-constrained least squares approach is useful for resolving such over-constrained situations. In our framework, the rotations of mean curvature normals are treated using the logarithms of unit quaternions in $\mathbb{R}^3$. During deformation, mean curvature normals can be rotated while preserving their magnitudes. In addition, we introduce a new modeling constraints called rigidity constraints and show that rigidity constraints can effectively preserve the shapes of feature regions during deformation. Our framework achieves good performance for interactive deformation of mesh models.

Key words: equality-constrained least squares, mesh deformation, quaternion, Laplacian coordinate, rigidity constraint

1 Introduction

Mesh deformation is useful in a variety of applications in computer graphics and computer-aided design. In recent years many discrete deformation techniques based on differential properties have been published [1–5]. They represent the differential

[*] Corresponding author. Tel: +81 3 5841 6511; fax: +81 3 5841 6511.
Email address: masuda@nakl.t.u-tokyo.ac.jp (H. Masuda).

properties of a given surface as a linear system and deform the surface so that the differential properties are preserved. In typical interactive mesh editing, the user first selects the region to be fixed, and then the vertices to be moved as the manipulation handle. Such user-specified conditions on vertices are called positional constraints. When the user drags the positions of handle vertices on the screen, the surface is deformed according to the handle manipulation.

Meyer et al. [6] approximated the mean curvature normals using cotangent weights. This method can produce good results even when triangles are not uniformly constructed. When a linear system is solved using the least squares method, positional constraints are satisfied by placing large weights in the least squares system. However, it is not easy to predict weight values so that positional constraints are satisfied within the allowable margin of error, because the magnitude of a mean curvature normal may be largely different at each vertex. As the values of weights on positional constraints become increasingly large, the solver satisfies positional constraints more strictly. However, very large weights make the solver numerically unstable when the mean curvature normals are represented using cotangent weights.

We propose a novel equality-constrained least squares approach in this paper. In our method, positional constraints are described as hard constraints. Then the combination of hard and soft constraints is converted to a typical least squares system using QR decomposition. This method stably and efficiently computes vertex positions that satisfy positional constraints precisely and preserve mean curvature normals in the least squares sense.

A well-known problem of hard constraints is over-constraints. If over-constraints are involved, the solver may halt the computation. Our framework is useful for resolving such over-constrained situations, which include redundant and conflicting constraints.

When handle vertices are rotated, mean curvature normals need to be rotated automatically[1]. We introduce a new rotation-propagation method that interpolates the logarithms of unit quaternions. Our method realizes the spherical linear interpolation of rotations from fixed vertices.

In addition, we propose a new modeling constraints called *rigidity constraints,* which preserve the shapes of feature regions during deformation. Rigidity constraints are very useful not to deform important features, such as eyes.

Our main contribution in this paper is as follows:

- A new deformation framework by means of the equality-constrained least squares method, which
  · incorporates hard constraints as well as soft constraints,
  · robustly calculates cotangent weights for Laplacian discretization, and
  · can detects conflicting and redundant constraints in hard constraints.

- A new rotation-propagation method by means of the interpolation of quaternion logarithms.
- Introduction of rigidity constraints that preserve the shapes of feature regions during deformation.

In the following section, we review the related work on 3D shape deformation. In Section 3, we describe our mesh deformation framework using the equality-constrained least squares method. In Section 4, we introduce rigidity constraints for feature regions. In Section 5, we will show how to manage over-constraints in hard constraints. In Section 6, we evaluate our framework and show experimental results. We conclude the paper in Section 7.

2 Related Work

Interactive mesh editing techniques have been intensively studied [7]. Such research aims to develop modeling tools for intuitively modifying free-form surfaces while preserving the geometric details.

There are several types of approach for mesh editing: free-from deformation (FFD), multiresolution mesh editing, and partial differential equation (PDE)-based mesh editing.

FFD methods are very popular approaches. They modify shapes implicitly by deforming 3D space in which objects are located [8–10]. However, it is difficult to manage geometric constraints defined on vertices, edges and faces, because FFD does not directly work on geometric shapes.

Multiresolution approaches [11–15] decompose a surface into a base mesh and several levels of details. Mesh editing can be performed at various resolutions. Botsch et al. [2,16] 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.

PDE-based approaches directly deform the original mesh so that the differential properties are preserved. The positions of the handle and fixed vertices are treated as boundary conditions during the editing processes. Laplacian operators are most commonly used to represent differential properties.

PDE-based approaches are categorized as non-linear and linear methods. Non-linear methods solve Laplacian or Poisson equations using non-linear iterative solvers [17–21]. These methods produce fair surfaces, but they are time-consuming and it is difficult to deform shapes interactively.

Linear PDE-based approaches encode Laplacian vectors and positional constraints in a linear system and obtain the deformed shapes by solving the linear system [1–5]. A discrete Laplacian operator is defined at each vertex by the weighted sum of difference vectors between the vertex and its adjacent neighbors. When the weights in the sum are represented using the Voronoi area and cotangents, the Laplacian vector approximates the mean curvature normal at the vertex [6].

Yu et al. [3] introduced a similar technique called Poisson editing, which manipulates the gradients of the coordinate functions (x,y,z) of the mesh. The vertex positions are calculated by solving discrete Poisson equations. Zhou et al. [5] proposed volumetric Laplacians to preserve the volumetric properties for large deformations. Nealen et al. [4] introduced a sketch-based interface on the Laplacian framework.

Since Laplacian vectors are defined in the local coordinate systems [1,22], one or more vertices must be specified in the global coordinate system to determine all vertex positions. Therefore, the total number of Laplacian vectors and positional constraints is greater than that of the unknowns. Sorkine et al. [1] solved this over-constrained problem approximately as a least squares system.

Several authors have discussed methods for rotating Laplacian vectors according to the deformation of surfaces. Lipman et al. [23] estimated the local rotations on the underlying smooth surface. Sorkine et al. [1] approximated rotations as linear forms and solved them using the least squares method. Lipman et al. [24] also proposed a rotation-invariant method, which encoded rotations and positions using relative positions on local frames and solved them as two separate linear systems. Zayer et al. [25] introduced a harmonic scalar function with range $[0,1]$ and specified a single unit quaternion at handle vertices. Then four components of unit quaternions for free vertices are interpolated using the weight values defined by the harmonic function.

Our approach to the rotation of mean curvature normals is similar to Zayer's method, although we assign the logarithms of unit quaternions to vertices and interpolate rotations using an energy-minimization surface in $\mathbb{R}^3$.

3 Framework Using Equality-Constrained Least Squares

3.1 Preliminaries

Let $M = (V,E,F)$ be a given triangular mesh with n vertices. V, E and F are the set of vertices, edges and faces, respectively. Vertex $i \in V$ has a three-dimensional coordinate $\mathbf{p}_i$. The original position of $\mathbf{p}_i$ is referred to as $\mathbf{p}_i^0$.

A discrete Laplacian operator $L(\mathbf{p}_i)$ and the original Laplacian vector δ_i are defined as:

$$
\begin{aligned}
L(\mathbf{p}_i) &= \sum_{j \in N(i)} w_{ij}(\mathbf{p}_i - \mathbf{p}_j), && (1)\\
\delta_i &= \sum_{j \in N(i)} w_{ij}(\mathbf{p}_i^0 - \mathbf{p}_j^0), && (2)
\end{aligned}
$$

where $N(i) = \{j | (i,j) \in E\}$ is the set of immediate neighbors of vertex i. Then the detail shape of a mesh model is encoded as the following linear equations:

$$
L(\mathbf{p}_i) = \delta_i. \tag{3}
$$

We describe positional constraints on the positions of a vertex, a point on an edge and a point on a face as follows:

$$
\begin{aligned}
&\mathbf{p}_i = \mathbf{u}_i, && (4)\\
&t\mathbf{p}_i + (1-t)\mathbf{p}_j = \mathbf{u}_{ij} && (5)\\
&s\mathbf{p}_i + t\mathbf{p}_j + (1-s-t)\mathbf{p}_k = \mathbf{u}_{ijk}, && (6)
\end{aligned}
$$

where $\mathbf{u}_i, \mathbf{u}_{ij}, \mathbf{u}_{ijk} \in \mathbb{R}^3$ are certain constant positions; t and s are scalar variables that satisfy $0 < t < 1$, $0 < s < 1$ and $0 < t + s < 1$.

When the combination of Eq. (3)-(6) are represented as the following linear system:

$$
A\mathbf{p} = \mathbf{c}, \tag{7}
$$

the least squares system can be described as:

$$
A^T A\mathbf{p} = A^T \mathbf{c}. \tag{8}
$$

Since $A^T A$ is a sparse symmetrical positive definite matrix, Eq. (8) can be very efficiently solved using sparse linear solvers [26–28]. After the matrix is factorized once, $\mathbf{p}$ can be repeatedly calculated according to the positions of the handle vertices.

The method for determining w_{ij} in Eq. (1) is important for numerical stability and quality. Two weighting methods have been commonly used for discrete Laplacian operators. A simple method is the uniform weights [1,4,17]:

$$
w_{ij} = \frac{1}{n_i}, \tag{9}
$$

where n_i is the number of vertices in $N(i)$. The other method is the cotangent weights [2,6,25,29]:

$$
w_{ij} = \frac{1}{2Area(i)}(\cot\alpha_{ij} + \cot\beta_{ij}), \tag{10}
$$

where $Area(i)$ is the Voronoi area of vertex i, and α_{ij} and β_{ij} are the angles opposite to edge (i,j), as shown in Figure 1. When Laplacian operators are defined using cotangent weights shown in Eq. (10), they are called Laplace-Beltrami operators [6]. A Laplace-Beltrami operator approximates the local normal direction and the local mean curvature at each vertex.

Even when badly shaped long triangles exist in a mesh, cotangent weights produce good results, although uniform weights may produce distorted shapes, as shown in Figure 2. Therefore, we use cotangent weights defined as Eq. (10) in our deformation framework.

However, when cotangent weights are used in a linear system, it is not easy to determine the weights of constraints so that positional constraints are satisfied within the allowable margin of error, because the magnitude of a mean curvature normal may be largely different at each vertex. In addition, when the mesh contains triangles that are too long or too small, large weights on positional constraints may cause numerical problems. Figure 3 shows such an example. Since this mesh has many badly shaped triangles, some linear solvers fail to calculate deformed shapes.

We need to remove this side effect of cotangent weights. One solution is to remesh models so that each triangle has the nearly same Voronoi area [16]. However, remeshing may change the details of shapes. Instead, we introduce hard constraints in the deformation framework and solve them using the equality-constrained least squares method. In our experiments, the numerical calculation is more stabilized by treating positional constraints as hard constraints. In addition, our framework is suitable for applications in which positional constraints should be precisely satisfied.

3.2 Equality-Constrained Least Squares

We represent soft constraints as a linear system:

$$A\mathbf{x} = \mathbf{c}, \tag{11}$$

and hard constraints as:

$$B\mathbf{x} = \mathbf{d}, \tag{12}$$

where $A \in \mathbb{R}^{l\times n}$, $B \in \mathbb{R}^{m\times n}$, $\mathbf{c} \in \mathbb{R}^{l}$, $\mathbf{d} \in \mathbb{R}^{m}$, $m \le n \le l$.

Then our mesh deformation framework with hard constraints can be formalized as:

$$\min_{B\mathbf{x}=\mathbf{d}} \|A\mathbf{x} - \mathbf{c}\|_2. \tag{13}$$

We solve this equation using the equality-constrained least squares method [30]. For clarity, in this section we assume that matrix B has full rank. Then B^T can be decomposed by QR decomposition [30] as:

$$B^T = QR, \tag{14}$$

where $Q \in \mathbb{R}^{n\times n}$ is an orthogonal matrix; $R \in \mathbb{R}^{n\times m}$ is an upper triangular matrix.

We rewrite these matrices as:

$$Q = \begin{array}{c} \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \\ \quad m \;\; n-m \end{array} \tag{15}$$

$$R = \begin{array}{c c} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} & \begin{array}{c} m \\ n-m \end{array} \end{array} \tag{16}$$

$$Q^T\mathbf{x} = \begin{array}{c c} \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} & \begin{array}{c} m \\ n-m \end{array} \end{array} \tag{17}$$

Thus, Eq. (13) is expressed in the following form:

$$\min_{R_1^T\mathbf{y}=\mathbf{d}} \|AQ_1\mathbf{y} + AQ_2\mathbf{z} - \mathbf{c}\|_2. \tag{18}$$

Since $\mathbf{y}$ is determined by the constraint equation $R_1^T\mathbf{y} = \mathbf{d}$, $\mathbf{z}$ is obtained by solving the following unconstrained least squares problem:

$$\min_{\mathbf{z}} \|AQ_2\mathbf{z} - (\mathbf{c} - AQ_1\mathbf{y})\|_2. \tag{19}$$

By solving this least squares system, we obtain:

$$(AQ_2)^T AQ_2\mathbf{z} = (AQ_2)^T(\mathbf{c} - AQ_1\mathbf{y}). \tag{20}$$

Since the size of $(AQ_2)^T(AQ_2)$ is smaller than A, we can solve Eq. (20) efficiently using sparse symmetrical positive definite linear solvers [27,28]. Finally, we can obtain $\mathbf{x}$ as:

$$\mathbf{x} = Q\begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix}. \tag{21}$$

We will note that hard constraints can be also solved by Lagrange multipliers[31]. However, since the Lagrange multipliers method produces a non-positive definite linear system with the fourth powers of coefficients, it is much more unstable than the equality-constrained least squares method. Therefore, the Lagrange multipliers method should be more carefully treated when cotangent weights are incorporated.

We will also note that our method may become slow when a large number of hard constraints contain two or more variables on the left-hand sides of equations, as shown in Eq.(5) and (6). This is because matrix AQ_2 in Eq.(20) becomes less sparse in such a case. In future work, we would like to investigate methods for stably and efficiently solving a large number of hard constraints with two or more variables.

3.3 Rotation of Mean Curvature Normals

When handle vertices are rotated, Laplacian vector δ_i in Eq. (3) should also be rotated. Figure 4 shows deformed shapes and the directions of mean curvature normals. When the original mean curvature normals are used to calculate vertex positions, deformed shapes may be distorted, as shown in the middle of Figure 4. The bottom figure shows a deformed shape in which mean curvature normals are rotated according to the rotation of handle vertices.

Shoemake [32] proposed the spherical linear interpolation between two rotations using unit quaternions. Johnson [33] applied the spherical linear interpolation to multiple unit quaternions using the logarithms of unit quaternions. We use a similar approach and assign the logarithms of unit quaternions to all vertices for calculating the rotation matrices of mean curvature normals.

A quaternion can be written in the form:

$$Q = (w,x,y,z) = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}, \tag{22}$$

where $w,x,y,z \in \mathbb{R}$; $\mathbf{i},\mathbf{j},\mathbf{k}$ are distinct imaginary numbers. When a quaternion has unit magnitude, it corresponds to a unique rotation matrix [32]. A unit quaternion can be represented using rotation axis $\hat{\mathbf{n}}$ and rotation angle θ as:

$$\hat{Q} = e^{\hat{\mathbf{n}}\frac{\theta}{2}} = cos\frac{\theta}{2} + \hat{\mathbf{n}}sin\frac{\theta}{2}, \tag{23}$$

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

$$\mathbf{q} = ln\hat{Q} = \frac{\theta}{2}\hat{\mathbf{n}}. \tag{24}$$

If $\mathbf{q}$ is given, the rotation axis and angle are calculated as $\hat{\mathbf{n}} = \mathbf{q}/|\mathbf{q}|$ and $\theta = 2|\mathbf{q}|$.

Then we assign logarithm $\mathbf{q}_i \in \mathbb{R}^3$ to vertex i. When the normal of vertex i is fixed as the original direction, $\mathbf{q}_i$ is equal to 0; when it is rotated around $\hat{\mathbf{n}}_i$ by angle θ_i, $\mathbf{q}_i$ is specified as in Eq. (24).

We interpolate the logarithms of free vertices using the ones of fixed and handle vertices. Pinkall and Polthier [29] proposed an interpolation technique using discrete conformal mapping. Zayer et al. [25] applied discrete conformal mapping for interpolating unit quaternions.

We introduce similar constraints on the logarithms of unit quaternions as:

$$L(\mathbf{q}_i) = \frac{1}{2Area(i)} \sum_{j \in N(i)} (cot\alpha_{ij} + cot\beta_{ij})(\mathbf{q}_i - \mathbf{q}_j) = 0. \tag{25}$$

We assign rotational constraints on fixed vertices as:

$$\mathbf{q}_i = 0, \tag{26}$$

on handle vertices as:

$$\mathbf{q}_i = \mathbf{r}_i, \tag{27}$$

where $\mathbf{r}_i$ is a certain constant vector in $\mathbb{R}^3$.

Eq. (25)-(27) construct a sparse linear system for rotations, which can be solved using sparse direct solvers in the same way as for positions. The solution of the linear system generates an energy-minimization surface [29] in three dimensional space spanned by the logarithms of unit quaternions.

When all logarithms of unit quaternions are calculated, a rotation matrix is uniquely determined at each vertex. The rotation matrices are applied to mean curvature normals δ_i in Eq. (3).

Figure 5 shows the example of a deformed shape, in which the mean curvature normals are rotated using the logarithms of quaternions.

Figure 6 visualizes the unit quaternions of vertices. In this figure, $\mathbb{S}^3$ for unit quaternions is reduced to $\mathbb{S}^2$ by restricting the coefficients of imaginary number $\mathbf{k}$ to 0, as in [34]. When all handle vertices rotate as a single rigid region, the unit quaternions of free vertices are located on the shortest arc of the sphere, as shown in the left of Figure 6; when more than two handles are specified and they are rotated differently, unit quaternions are located inside the region constructed by arcs between the unit quaternions of fixed and handle vertices, as shown in the center and right of Figure 6;

In our method, each vertex in handles can have a different rotation matrix and therefore handles do not have to be rigid regions, although the method in [25] is restricted to the rotation of a single rigid handle.

It should be noted that our rotation method needs to explicitly rotate the handle vertices, as well as the method in [24]. In contrast, the methods in [1,23] rotate local normals in implicit manners. In our current implementation, two methods are provided to the user for rotating handle regions. One method places the rotation center at the center of the fixed region and automatically rotates the handle region by the angle between the current and original handle positions on the screen. The other method requires the user to explicitly rotate the local coordinate system on the handle region. However, this method leaves to the user the responsibility for maintaining the compatibility of positions and rotations[24]. In future work, we would like to develop a variety of rotation methods for handle regions.

4 Rigidity Constraints for Feature Regions

In addition to positional constraints, we introduce rigidity constraints that constrain the relative positions of vertices as:

$$\mathbf{p}_i - \mathbf{p}_j = \mathbf{u}_{ij}, \tag{28}$$

where $\mathbf{u}_{ij} \in \mathbb{R}^3$ is a certain constant vector. These constraints are added to a linear system defined in Eq.(7).

When rigidity constraints are specified to edges in a feature region, the shape of the feature region is preserved as a rigid body while deformation. The user can specify such a feature region by drawing a closed curve on a mesh model. Then the spanning tree is traversed in the feature region and its edges are constrained using rigidity constraints.

Since a feature region has to be rotated as a rigid body according to the rotation of the handle region, we add the following constraints in a linear system for rotations:

$$\mathbf{q}_i - \mathbf{q}_j = 0, \tag{29}$$

where (i, j) are all pairs of vertices to which rigidity constraints are specified. When the logarithms of unit quaternions are calculated, vector $\mathbf{u}_{ij}$ in Eq. (28) is rotated around axis $\mathbf{q}_i/|\mathbf{q}_i|$ by angle $2|\mathbf{q}_i|$.

Constraints on feature regions are useful to preserve important features in mesh models. Figure 7 shows the effectiveness of our method. In this example, feature regions are specified at the eyes of a mannequin model. While the eyes in Figure 7(c) are stretched and blurred, their shapes in Figure 7(d) are the same as the original ones.

It is also possible to preserve the distance between specified positions. In Figure 8, the distance between the mouse and the nose is preserved by specifying a rigidity constraint.

5 Resolution of Over-Constraints

In our framework, positional constraints are strictly satisfied as hard constraints. Therefore, if conflicting or redundant constraints are involved, they lead to the rank deficiency in matrix B, and the solver may halt the computation.

We can resolve such over-constraint problems by detecting and removing the deficiency of the rank during the processes of QR decomposition.

QR decomposition $B^T = QR$ can be computed using Householder factorization [30]. Each column in B^T corresponds to a hard constraint and it is processed sequentially. When the degree of freedom (DoF) of vertex positions are n at the beginning of decomposition, the DoF is reduced to $n-i$ after the process of column i is completed. If B^T has no redundant constraint, the DoF of vertices becomes $n-m$ after QR decomposition.

If column j in B^T is a redundant constraint, we cannot decrease the degree of freedom by processing column j, because the diagonal and lower elements of column j are equal to zero after the previous $j-1$ columns are processed. Therefore, we can detect the redundant constraints.

When the column of a redundant constraint is detected, it is skipped and then the next column is processed. By removing all the redundant constraints, a unique solution is determined using the unconstrained least squares method.

After detecting all redundant constraints, we examine whether each constraint satisfies the solution. If a redundant constraint is consistent with the solution, it is simply ignored. If it conflicts with the solution, we can send out a warning message to correct the specification.

6 Experimental Results

In this section, we show some results of deformation based on the equality-constrained least squares method.

Figure 9 shows examples of deformed shapes. This mesh has badly shaped triangles, as shown in Figure 3. Our method produced good results.

Figure 10 shows sample models that have been commonly used for evaluation in computer graphics. We solved the linear systems using TAUCS [27], which is a well-known solver of Cholesky factorization.

Table 1 shows the numerical stability. When all constraints are treated as soft constraints and the weight values on mean curvature normals and positional constrains are specified as 1 and 5, respectively, the linear solver[27] failed to calculate the least squares systems of Bunny and Dragon models in our experiments. When equality-constrained least squares systems are constructed and solved using the same linear solver, they could be successfully solved in both cases.

Figure 11 shows a mesh model with conflicting constraints. In the left figure, conflicting positional constraints are assigned to the same vertices. In the center figure, all constraints are treated as soft constraints and a compromised shape is generated. In this case, it is difficult to detect which constraints are conflicting. The right figure shows that our method can resolve the conflicting constraints by detecting and removing them.

Our calculation process consists of three phases: matrix set-up, decomposition and solution. In the set-up phase, we construct the matrices for hard and soft constraints. In the decomposition phase, we obtain the least squares systems by QR decomposition and factorize them. In the solution phase, we compute the positions of vertices. After the matrix is factorized once in the decomposition phase, the vertex positions are repeatedly calculated in the time spent on the solution phase. A longer computation time is required in the set-up and solution phases than in the solution phase. Figure 12 compares the total time for the first two phases using the least squares method with only soft constraints and our method with hard and soft constraints. The calculation time was measured on a PC with 2.0-GHz Pentium-M CPU, with 1.0 GB of RAM and Windows OS. The result shows that our method with hard and soft constraints achieved almost equivalent performance compared to the method with only soft constraints, although our method requires the overhead of QR decomposition.

7 Conclusion

In this paper, we proposed an equality-constrained least squares approach for stably computing mesh deformation that strictly satisfies positional constraints and approximately preserves mean curvature normals. Our experimental results for mesh models showed that our method is sufficiently stable compared to existing methods based on the least squares method. The performance of our method was as good as that of the method that manages only soft constraints. We also showed that our framework is useful for resolving such over-constrained situations.

We also proposed a new method for rotating mean curvature normals using the logarithms of unit quaternions. We showed that this method produces good deformed shapes. In addition, we introduced rigidity constraints for preserving the shapes of feature regions and the distances between specified positions. This method is useful for preserving important feature regions in deformed shapes.

In future work, we need to tune up the numerical solves and improve the performance, since our current implementation is not optimized yet. We showed our experimental results that cause numerical stability. Although we removed the cause of large weights on positional constraints, but we think there are other causes of instability, such as very small angles of triangles and too small and too large Voronoi areas. We need to investigate causes of instability more carefully and find methods for remove them. In addition, we would like to develop intuitive tools for specifying constraints and handle vertices. It will be interesting to develop constraint-based modeling functions on our current framework. Since our method strictly satisfies hard constraints, it can be applied to applications that require precise positional constraints, such as engineering applications. We would also like to investigate requirements on engineering applications.

Acknowledgements

The Armadillo, Bunny and Dragon models are courtesy of Stanford University, and the Mannequin model is courtesy of the Graphics and Imaging Laboratory, University of Washington.

References

[1] O. Sorkine, Y. Lipman, D. Cohen-Or, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184. ACM Press, 2004.

[2] M. Botsch and L. Kobbelt. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph.*, 23(3):630–634, 2004.

[3] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.*, 23(3):644–651, 2004.

[4] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for detail-preserving mesh editing. *ACM Trans. Graph.*, 24(3):1142–1147, 2005.

[5] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum. Large mesh deformation using the volumetric graph laplacian. *ACM Trans. Graph.*, 24(3):496–503, 2005.

[6] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 35–57. Springer-Verlag, 2003.

[7] O. Sorkine. Laplacian mesh processing. In Y. Chrysanthou and M. Magnor, editors, *STAR Proceedings of Eurographics 2005*, pages 53–70. Eurographics Association, Sept. 2005.

[8] S. Coquillart. Extended free-form deformation: A sculpturing tool for 3d geometric modeling. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 187–196. ACM Press, 1990.

[9] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. In *SIGGRAPH '96: Proceedings of the 23nd annual conference on Computer graphics and interactive techniques*, pages 181–188, 1996.

[10] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160. ACM Press, 1986.

[11] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 173–182. ACM Press, 1995.

[12] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 325–334. ACM Press/Addison-Wesley Publishing Co., 1999.

[13] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 105–114. ACM Press, 1998.

[14] S. Lee. Interactive multiresolution editing of arbitrary meshes. *Comput. Graph. Forum*, 18(3):73–82, 1999.

[15] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *SIGGRAPH*, pages 259–268, 1997.

[16] M. Botsch and L. Kobbelt. A remeshing approach to multiresolution modeling. In *Symposium on Geometry Processing*, pages 189–196, 2004.

[17] G. Taubin. A signal processing approach to fair surface design. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358. ACM Press, 1995.

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

[19] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324. ACM Press/Addison-Wesley Publishing Co., 1999.

[20] R. Schneider and L. Kobbelt. Generating fair meshes with g1 boundary conditions. In *GMP*, pages 251–261, 2000.

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

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

[23] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel. Differential coordinates for interactive mesh editing. In *SMI 2004: Proceedings of the international conference on Shape Modeling and Applications*, pages 181–190, 2004.

[24] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph.*, 24(3):479–487, 2005.

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

[26] M. Botsch, D. Bommes, and L. Kobbelt. Efficient linear system solvers for mesh processing. In *IMA Conference on the Mathematics of Surfaces*, pages 62–83, 2005.

[27] S. Toledo, D. Chen, and V. Rotkin. Taucs: A library of sparse linear solvers. http://www.tau.ac.il/ stoledo/taucs/, 2003.

[28] N. I. M. Gould, Y. Hu, and J. A. Scott. 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.

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

[30] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins Series in the Mathematical Sciences. Johns Hopkins University Press, 3rd edition, 1989.

[31] W. Welch and A. Witkin. Variational surface modeling. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, vol. 26, 157–166, 1992.

[32] K. Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH '85: Proceedings of the 12nd annual conference on Computer graphics and interactive techniques*, pages 245–254, July 22–26 1985.

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

[34] E. B. Dam, M. Koch, and M. Lillholm. Quaternions, interpolation and animation. Technical Report DIKU-TR-98/5, Department of Computer Science, University of Copenhagen, 1998.

Table 1
Comparison of numerical stability. (×: failed to compute)

	Only soft constraints by unconstrained least squares		Soft & hard constraints by equality-constrained least squares	
Weighting	Uniform	Cotangent	Uniform	Cotangent
(a)Armadillo (Figure 10c)	○	○	○	○
(b)Bunny (Figure 10d)	○	×	○	○
(c)Dragon (Figure 9)	○	×	○	○

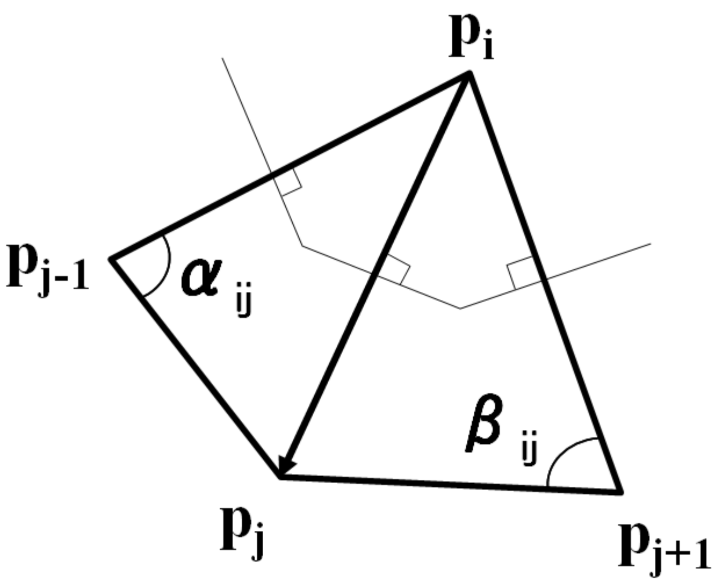


Fig. 1. Definition of α_{ij} and β_{ij}.

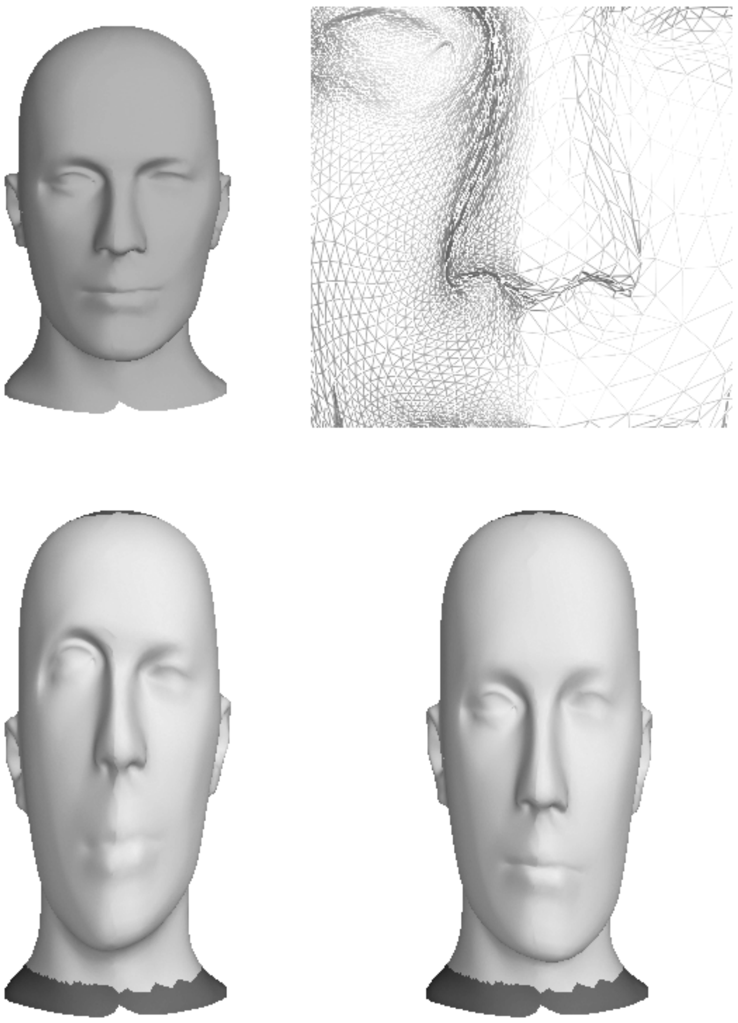


Fig. 2. Comparison of results with different weighting methods. Top: The original mesh subdivided unsymmetrically; bottom left: deformed with uniform weights; bottom right: deformed with cotangent weights.

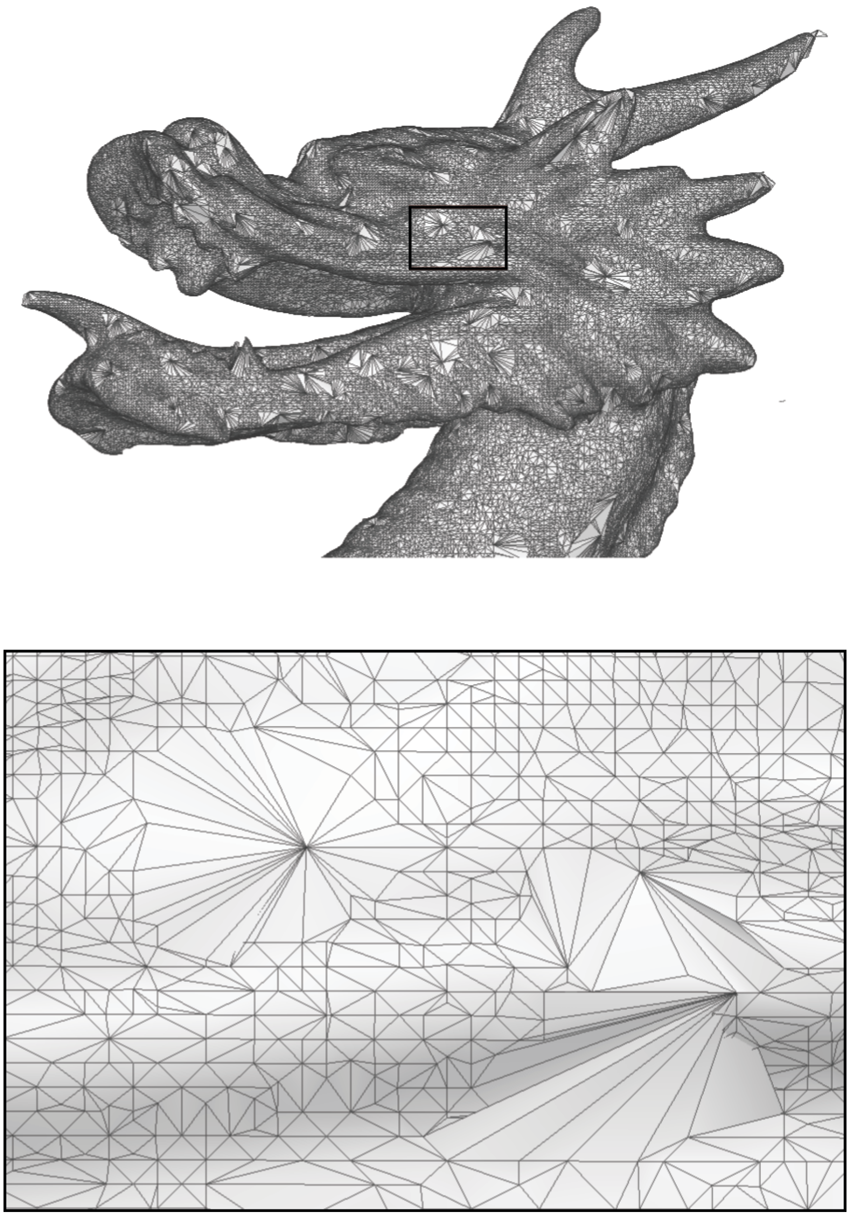


Fig. 3. Mesh model with badly shaped triangles.

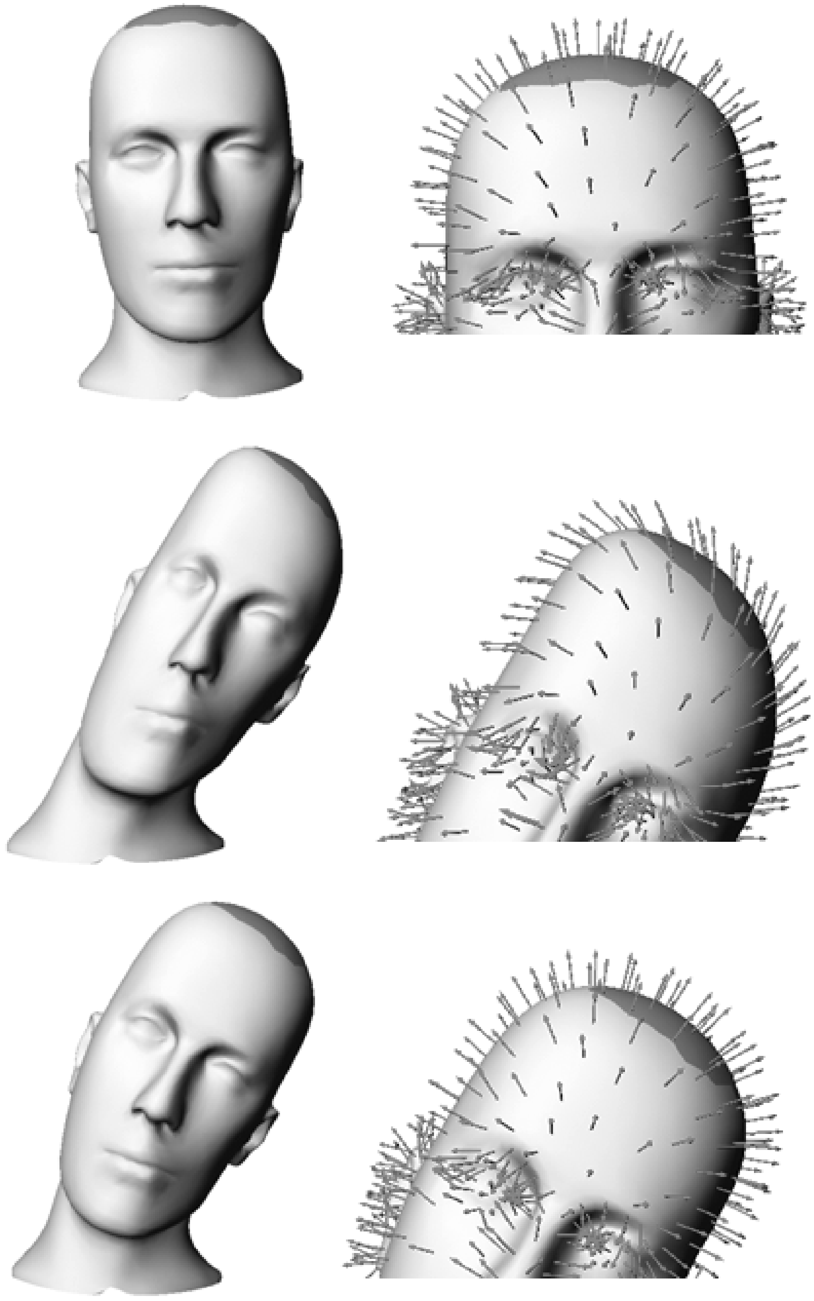


Fig. 4. Rotation of mean curvature normals. Top: original shape and its local normals; middle: deformed without rotation; bottom: deformed using rotated mean curvature normals.

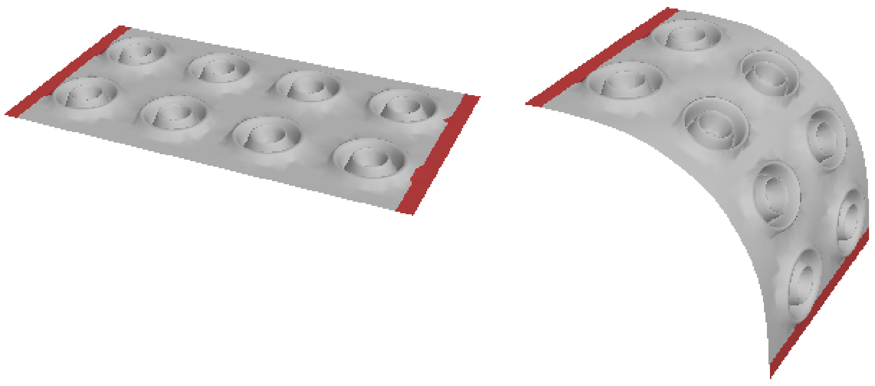


Fig. 5. Deformation using rotated mean curvature normals. Left: original shape; right: deformed shape.

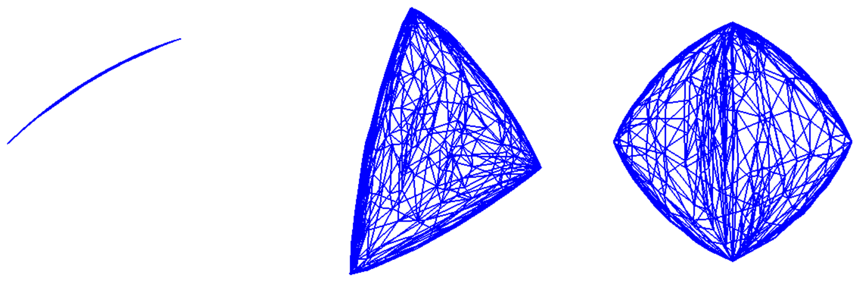


Fig. 6. Visualization of unit quaternions. Left: interpolated between two unit quaternions; middle: between three unit quaternions; right: between four unit quaternions.

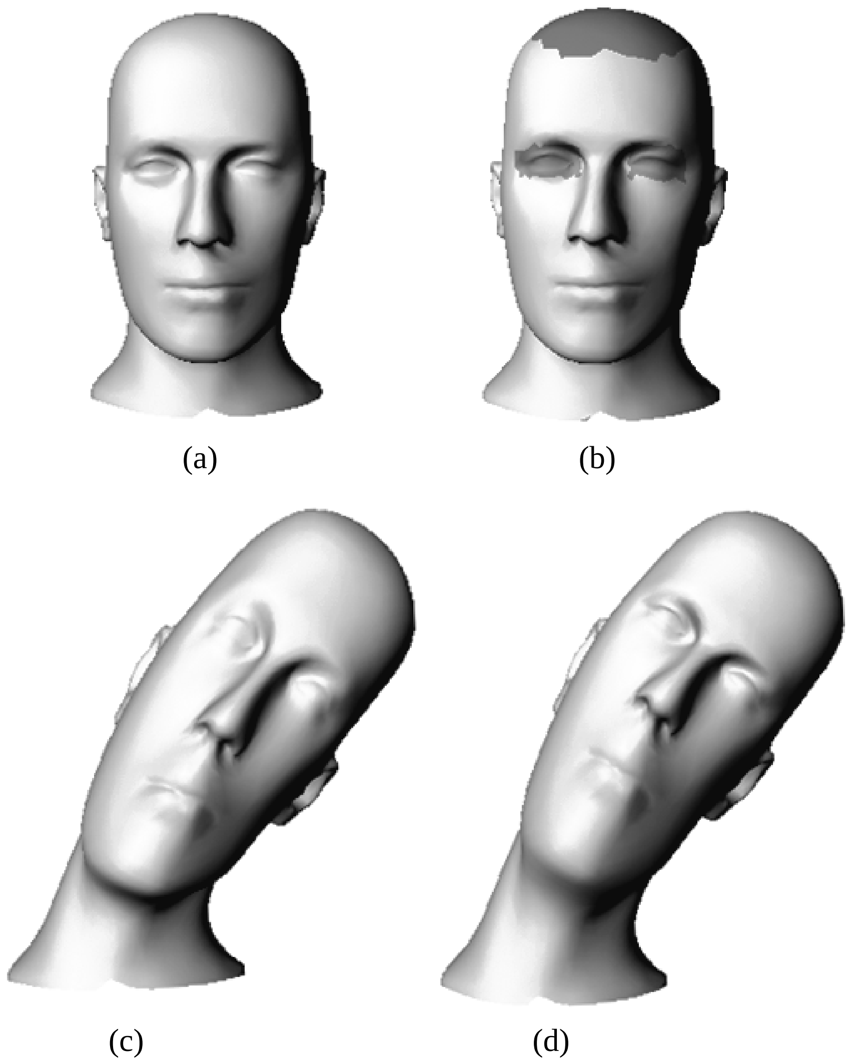


Fig. 7. Preservation of Feature Regions. (a) the original shape; (b) feature regions at the eyes and handle region at the head; (c) deformed without feature regions; (d) deformed while preserving feature regions.

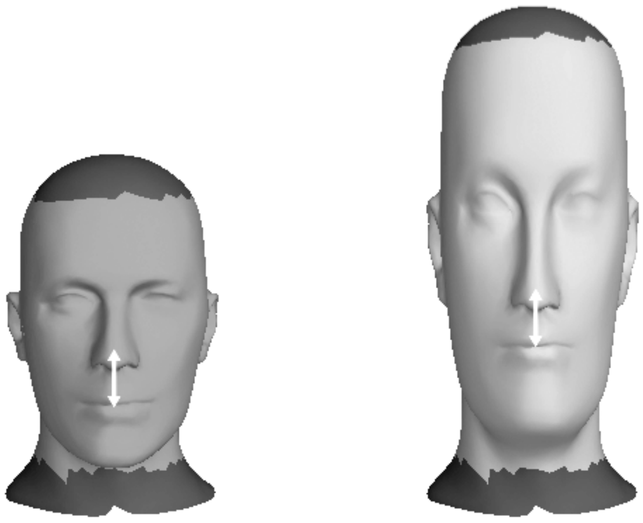


Fig. 8. Preservation of Distances. Right: constrained the distance between the mouse and the nose; left: deformed while preserving the distance.

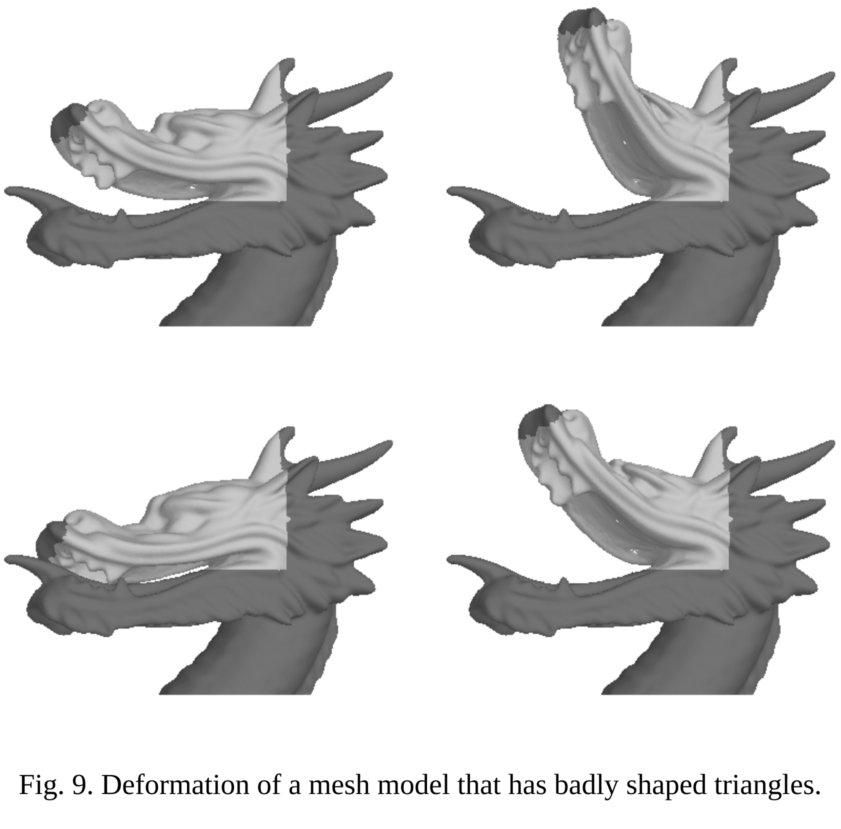
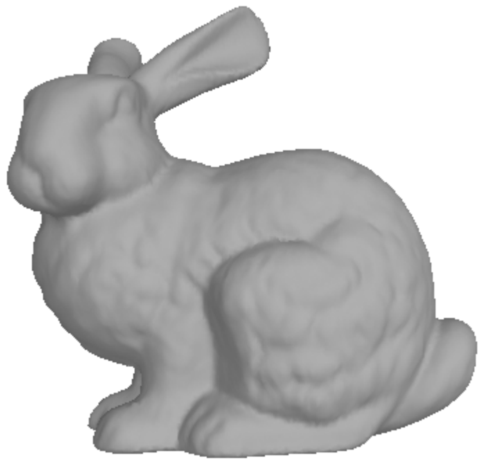
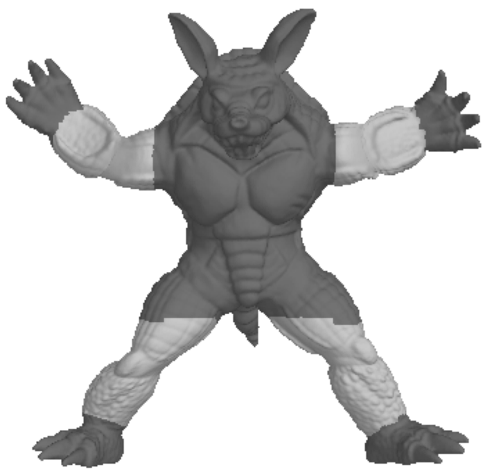


Fig. 9. Deformation of a mesh model that has badly shaped triangles.

(a)


(b)


(c)

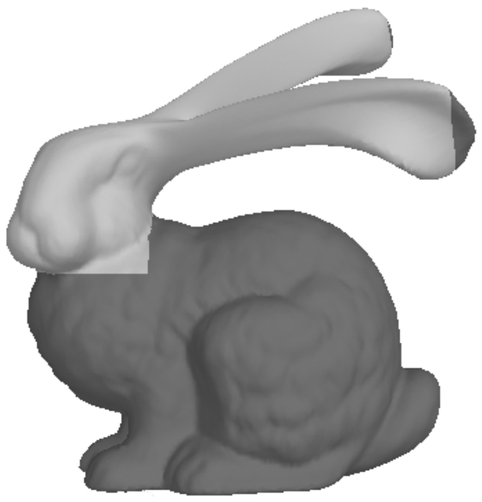

(d)

Fig. 10. Examples of deformed meshes.

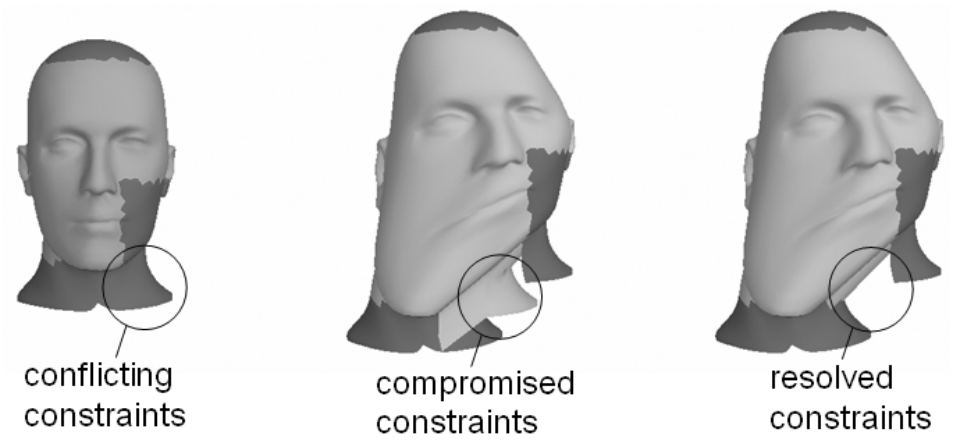


Fig. 11. Example of conflicting constraints. Left: the original mesh with conflicting constraints; center: compromised solution; right: result of conflict resolution.

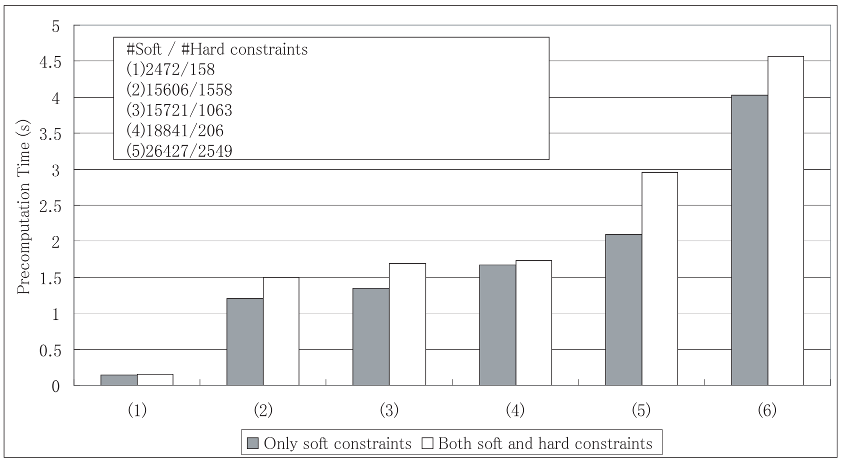


Fig. 12. Total computation time for the set-up and decomposition of matrices. Left bar: soft constraints solved by the least squares method; right bar: soft and hard constraints solved by the constrained least squares method.