Graphical Models 73 (2011) 10–19

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

Robust mesh editing using Laplacian coordinates

Shaoting Zhang *, Junzhou Huang, Dimitris N. Metaxas

Department of Computer Science, Rutgers University, Piscataway, NJ 08854, United States

ARTICLE INFO

Article history:
Received 30 April 2010
Received in revised form 12 October 2010
Accepted 15 October 2010
Available online 10 November 2010

Keywords:
Mesh editing
Laplacian coordinate
Skeleton
Volume preservation
Mesh optimization

ABSTRACT

Shape deformation and editing are important for animation and game design. Laplacian surface based methods have been widely investigated and used in many works. In this paper we propose a robust mesh editing framework which improves traditional Laplacian surface editing. It consists of two procedures: skeleton based as-rigid-as-possible (ARAP) shape modeling and detail-preserving mesh optimization. Traditional ARAP shape modeling relies on the mesh quality. Degenerated mesh may adversely affect the deformation performance. A preprocessing step of mesh optimization can alleviate this problem. However, skinny triangles can still be generated during deformation, which adversely affect the editing performance. Thus our method performs Laplacian mesh deformation and optimization alternately in each iteration, which ensures mesh quality without noticeably increasing computational complexity or changing the shape details. This approach is more robust than those solely using Laplacian mesh deformation. An additional benefit is that the skeleton-based ARAP modeling can approximately preserve the volume of an object with large-scale deformations. The volume is roughly kept by leveraging the skeleton information and employing a carefully designed energy function to preserve the edge length. This method does not break the manifoldness of traditional ARAP methods or sacrifice speed. In our experiments, we show that (1) our method is robust even for degenerated meshes, (2) the deformation is natural in terms of recovering rotations, and (3) volumes are roughly kept even under large-scale deformations. The system achieves real time performance for surface meshes with 7k vertices.

1. Introduction

Shape deformation is widely used in many applications such as animation, game design and virtual reality. During the past decade, people have put a lot of effort in as-rigid-as-possible (ARAP) shape modeling to obtain natural deformations. ARAP deformation means that the shape should be locally preserved, and the deformations are locally similar. This principal originates from the classical elastic energy that measures the difference between two shapes, or shell energy [25].

2D shape manipulation: Previous works try to constrain transformation matrices to achieve the ARAP effect. In 2D, there have been many excellent methods. Alexa et al. [1] successfully applied ARAP deformation to shape interpolation. Sorkine et al. [24] showed that the similarity transformation in 2D can be completely characterized with a linear expression. Igarashi and Moscovich [10] presented a two-step closed form algorithm to deform 2D shapes. The first step allows translation, rotation and uniform scaling. The second step adjusts the scaling locally. Schaefer et al. [18] devised a moving least squares framework for 2D space warping, where each element of the space grid deforms ARAP, and the warping is controlled by positional constraints on several grid points. These methods have shown promising performance on 2D ARAP deformations.

Differential mesh editing: 3D ARAP deformation is more challenging because of the nonlinearity of similarity transformations. Botsch and Kobblet [5] formulated a linearized system to represent ARAP deformation. It allows an efficient optimization, but causes artifacts such as local details and general shape distortions for large deformations. Zorin et al. [34] and Guskov et al. [7] proposed a multi-resolution

* Corresponding author.
E-mail address: shaoting@cs.rutgers.edu (S. Zhang).

1524-0703/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.gmod.2010.10.003

technique to deform low-frequency components of the surface first, followed by adding back high-frequency details as local displacements. However, this solution leads to local self-intersections when the deformation introduces bending. Sorkine et al. [24] proposed Laplacian surface editing (LSE), which uses a first-order approximation of similarity transformations. It works well when only moderate rotations are involved in the deformation. Dual LSE [2] starts with naive Laplacian editing as an initial guess and iteratively adjusts the local Laplacian coordinates to refit the surface geometry to those Laplacians. Propagating transformation [13,30,31] can also alleviate the rotation related problems. However, such editing methods are insensitive to translations and may lead to unintuitive shape distortions. Nonlinear ARAP approaches were also proposed, such as the volumetric graph Laplacian (VGL) [33], Laplacian constraints [8], PriMo [6], moving frames [12] and ARAP surface modeling [22], all of which produce compelling results. Particularly, ARAP Surface modeling uses an iterative scheme to minimize its carefully designed energy formulation, which is easy-to-implement and closely related to the widely used LSE. The rotations are natural and edge lengths are preserved, even for large-scale deformations. Its results are compared with those of Poission mesh editing [28,30], and the deformations are more natural.

The main shortcomings of existing ARAP surface modeling [22] are twofold. First, the performance relies on the mesh quality. The mesh may be degenerated in the beginning or during deformation, which adversely affects the robustness of the ARAP methods. Second, there is no volume preserving constraint in these surface modeling methods, which may cause undesired artifacts for large-scale deformation.

In this paper we propose a robust mesh editing framework based on Laplacian coordinates. This two-step framework consists of (1) a skeleton-based ARAP modeling, and (2) a detail-preserving mesh optimization. The first step is an efficient and easy-to-implement method to approximately preserve the volume by leveraging skeleton information. The energy formulation of [22] is adopted to generate natural rotations and to preserve edge lengths. In addition, a skeleton is generated and points on the skeleton are correlated with vertices on the surface. The connectivity of each cross section is defined as *skeleton edges* and is added into the traditional linear ARAP system as additional rows. Using the edge-length preserving property, the cross section area and volume can be roughly preserved. The second step is a remeshing technique to improve the mesh quality without affecting the shape details. Similar ideas of remeshing during deformation appear in stretch-based mesh manipulation [20], which incorporates the mesh quality metrics into the formulation. In our work, the Laplacian mesh optimization is directly used as a regularization step during deformation. Since both deformation and remeshing procedures require the solutions to similar large-scale sparse linear systems, sparse Cholesky factorization [26] is used as an initialization. Then the two linear systems are solved efficiently using back substitution. Using these two procedures alternately during deformation allows robust and efficient deformations of the mesh.

Volume preserving editing: Since there are similar approaches dealing with volume preservation or large-scale deformation, we first discuss the differences between our method and others, before listing our contributions. Volumetric graph Laplacian (VGL) [33] first constructs a graph representing the volume inside the input mesh. The Laplacian of this graph encodes the volumetric details as the difference between each vertex in the graph and its neighbors. Preserving such detail imposes a volumetric constraint. This model works well for large scale deformation and does not create intersection artifacts. However, a preprocessing step is needed to generate this graph, and the computation complexity is increased when introducing this relatively complex interior structure. Huang et al. [9] developed a subspace method that builds a coarse control mesh around the original one, and projects linear or nonlinear deformation constraints onto the control mesh using mean value interpolation. Using this subspace formed by this control mesh, the minimization is both fast and stable. Mesh Puppetry [19] is a variational approach which uses cascading optimization and inverse kinematics to perform large-scale mesh deformation. New poses are created by specifying constraints on the vertex positions, the balance of the character length, the rigidity preservation and the joint limits. Since many of these constraints are nonlinear, a novel cascading optimization procedure is proposed to solve the system in realtime (50K+vertices). The weights of these constraints are important factors and altering the weights can produce designed results or partially solve some issues. Au et al. [3] proposed an effective approach using handle-aware isolines for scalable shape editing. The isolines act as a generalized skeletal structure and are independent of mesh sampling. Thus this reduced model achieves detail-preserving deformation with resolution-independent cost per iteration, fast convergence rate and linear memory cost, and has excellent scalability. Our volume preservation method differs from the methods listed above, in that it maintains the volume magnitude by combining the benefits of an internal structure and the edge-length preserving property of the energy function of the ARAP surface modeling [22]. Thus it is compatible with theories of LSE or ARAP surface modeling methods, and can be easily built upon their existing systems.

Contributions: The contributions of our method are as follows: (1) We propose a two-step framework to robustly deform the mesh. Since both procedures rely on Laplacian coordinates, most of computations can be reused. This makes our method both robust and computationally efficient. (2) We propose an approach to integrate the skeleton information with ARAP surface modeling to approximately preserve the volume without breaking its manifoldness or adversely affecting the computation speed; (3) In terms of implementation, our approach is straightforward to add to existing modeling frameworks relying on LSE or ARAP surface modeling.

The rest of the paper is organized as follows. Section 2 details our algorithm, including a skeleton-based ARAP modeling method and a detail-preserving optimization technique. Section 3 shows the experimental results, which demonstrate the robustness and volume preserving property of our method. Section 4 concludes this paper.

2. Algorithms

2.1. Framework

Our algorithm framework consists of two procedures: (1) mesh deformation and (2) optimization. The mesh deformation recovers natural rotations and preserves the volume magnitude even under large-scale deformations. The optimization improves the mesh quality without changing the shape details. The two procedures are alternated in each iteration. The deformations gain in stability from the improved mesh quality, while the optimization procedure does not noticeably increase the computational complexity. The remaining sections detail the mesh deformation and optimization, and discuss implementation and acceleration considerations.

2.2. Mesh deformation

Linear LSE: Laplacian coordinates represent each point as the weighted difference between itself and its neighborhoods. Given the original coordinates ($\mathbf{V} = [v_1, \ldots, v_n]^T$), the connectivity, and m control points, the coordinates of the reconstructed object ($\mathbf{V}' = [v'_1, \ldots, v'_n]^T$) can be obtained by minimizing the quadratic energy function:

$$\|\mathbf{L}\mathbf{V}' - \boldsymbol{\Delta}\|_2^2 + \Sigma_{i=1}^{m} |v'_{c_i} - v_{c_i}|^2 \tag{1}$$

where $\mathbf{L}$ represents the discrete Laplace–Beltrami operator (Fig. 1) using uniform ($\mathbf{L}_u$) or cotangent weights ($\mathbf{L}_c$) [17,14], $\boldsymbol{\Delta} = \mathbf{L}_c\mathbf{V}$ (computed beforehand), and v_c denotes the control points. The first term penalizes the shape difference after reconstruction, and the second term penalizes the change of positions of the control points. With m control points, (1) can be minimized as the $(n+m)\times n$ overdetermined linear system:

$$\begin{bmatrix} \mathbf{L} \\ \mathbf{I}_c \end{bmatrix} \mathbf{V}' = \begin{bmatrix} \boldsymbol{\Delta} \\ \mathbf{V}_c \end{bmatrix} \tag{2}$$

where $\mathbf{I}_c$ is the index matrix of $\mathbf{V}_c$, which maps each $\mathbf{V}'_c$ to $\mathbf{V}_c$. The reconstructed shape looks generally natural when rotations are small.

Rotation and edge constraints: Sorkine and Alexa [22] introduced a nonlinear approach to find natural rotations, which consists of two-steps. In the first step, an initial guess is calculated by linear LSE (or defined as the previous

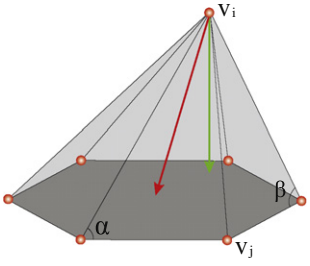

Fig. 1. Vertex v_i and its 1-ring neighbors. The red arrow (center) is the vector obtained from the uniform weights, which points to the centroid. The green arrow (to the right) is the vector obtained from the cotangent weights ($\frac{1}{2}(\cot\alpha + \cot\beta)$), which approximates the normal.

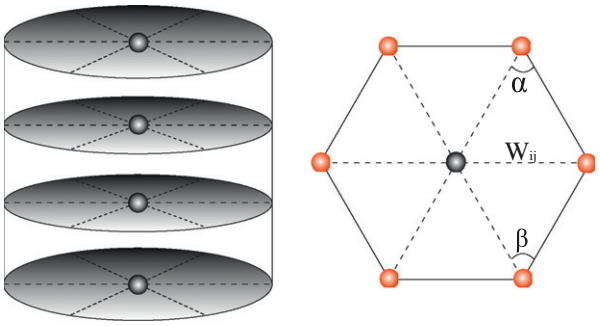

Fig. 2. Left: the skeleton points (black spheres) are inside the cylinder. Each point is connected to surface vertices by skeleton edges. Right: a cross section view, corresponding to a grey disk on the left. The skeleton edges are represented by dashed lines. Surface vertices, shown as the red spheres on the edges of the polygon, are on the cylinder's surface. W_{ij} is the cotangent weight of the skeleton edge below it, calculated as $W_{ij} = \frac{1}{2}(\cot\alpha + \cot\beta)$ [14].

frame). In the second step, a rotation matrix R_i is computed for each vertex by minimizing:

$$\Sigma_{j\in\mathbb{N}(i)} \|(v'_i - v'_j) - R_i(v_i - v_j)\|_2^2 \tag{3}$$

where $\mathbb{N}(i)$ represents the set of neighbors of the ith vertex and R_i is the rotation matrix for the ith vertex and depends on its neighbors. Minimizing (3) amounts to minimizing the change of edge lengths. Denoting the edge $e_{ij} = v_i - v_j$, one can write the covariance matrix S_i as:

$$S_i = \Sigma_{j\in\mathbb{N}(i)}(w_{ij} e_{ij} e_{ij}^{T\prime}) \tag{4}$$

where w_{ij} is the cotangent weight of e_{ij}, and e'_{ij} is the edge after reconstruction. The rotation matrix R_i of the ith vertex is derived from the singular value decomposition (SVD) of the covariance matrix $S_i = U_i\Sigma_i V_i^T$, and $R_i = V_i U_i^T$ [22,21]. A new linear system is then obtained by plugging R_i into the righthand side of (2) based on the derivative of (3) that ensures convergence [22]. The two procedures can be alternately performed to recover natural rotations when there is no large stretching. However, there is no volume preserving constraint in this method.

Skeleton and volume constraints: Volume preservation is important in many real-world applications. By using the volumetric mesh, the volume magnitude can be preserved to some degree. However, the volumetric mesh conflicts with traditional ARAP and Laplacian methods, which are designed for 2D manifolds. Furthermore, it increases the computation complexity. In our method, the skeleton information is incorporated to approximately keep the volume without breaking the manifoldness or significantly increasing the computational complexity.

To initialize the model, two-steps are needed. In the first step, a skeleton is manually defined or automatically generated using mesh contraction [4], and points are evenly generated from the skeleton (black spheres in Fig. 2). Since our focus is on the deformation, robust and automatic skeleton extractions are left for future investigation. In the second step, each skeleton point is correlated with vertices on the surface (red[1] points in Fig. 2). In this

[1] For interpretation of color in all figures, the reader is referred to the web version of this article.

step, rays perpendicular to the skeleton segment are emitted from each skeleton point and intersected with the surface. The surface vertices closest to these intersections are connected to that skeleton point. The connections between skeleton points and surface vertices are defined as *skeleton edges* (dashed lines in Fig. 2). Skeleton edges corresponding to the same skeleton point define a cross section (grey disks in Fig. 2). Although the object's topology changes after adding skeleton edges, the surface and each cross section are still 2D manifolds, which can be easily added to the ARAP framework. Note that there is no skeleton edge between skeleton points, since this would break the manifoldness.

By minimizing (3), the optimization strives to converge to a state where the edge length error is small if the modeling constraints do not impose large stretching on the surface. Because of the length preservation of both skeleton and surface edges, the areas of triangles consisting of these edges are kept when skeleton edges are relatively dense. Thus areas of cross sections consisting of these triangles are approximately the same. By combining the areas with surface edges connecting two cross sections, the volume in-between can be roughly kept unchanged. Based on the same idea, the volume of the main object is approximately preserved during deformations, to the extent allowed by the constraints (3). To add these skeleton edges as new constraints, we append additional rows to (2):

$$\begin{bmatrix} \mathbf{L} & | & \mathbf{0} \\ \mathbf{I}_c & | & \mathbf{0} \\ & \mathbf{L}_s & \end{bmatrix} \begin{bmatrix} \mathbf{V}' \\ \mathbf{V}'_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Delta} \\ \mathbf{V}_c \\ \boldsymbol{\Delta}_s \end{bmatrix} \tag{5}$$

where $\mathbf{L}_s$, $\mathbf{V}'_s$ and $\boldsymbol{\Delta}_s$ are the uniform Laplace–Beltrami operator, cartesian coordinates and Laplacian coordinates of skeleton points, respectively. Assuming there are l skeleton points and m control points, then (5) is a $(n+m+l)\times(n+l)$ overdetermined linear system. Since $\mathbf{L}_s$ depends on the connectivity and relative positions between skeleton points and surface vertices, it is multiplied by both $\mathbf{V}'$ and $\mathbf{V}'_s$ to compute $\boldsymbol{\Delta}_s$, while $\mathbf{L}$ is only multiplied by $\mathbf{V}'$ when computing $\boldsymbol{\Delta}$. In other words, $\mathbf{V}'_s$ is coupled to $\mathbf{V}'$, but not vice versa. This setting ensures that the first $(n+m)$ rows are the same as (2), with l additional zero columns. Thus the shape and control point constraints are the same. The newly added l rows naturally incorporate the volume preserving constraint without breaking the original system. In the same way in which (2) corresponds to (1), solving (5) amounts to -minimizing the following energy function:

$$\|\mathbf{L}\mathbf{V}' - \boldsymbol{\Delta}\|_2^2 + \|\mathbf{L}_s\mathbf{V}'_{all} - \boldsymbol{\Delta}_s\|_2^2 + \Sigma_{i=1}^{m}|v'_{c_i} - v_{c_i}|^2 \tag{6}$$

where $\mathbf{V}'_{all}$ contains all surface vertices and skeleton points. Thus adding these new constraints corresponds to appending a new term to the energy function.

Weighted least square can be used to adjust the importance among surface vertices, control points and skeleton points. In this case, the linear system (5) is rewritten as:

$$\begin{bmatrix} \mathbf{W}_L\mathbf{L} & | & \mathbf{0} \\ \mathbf{W}_c\mathbf{I}_c & | & \mathbf{0} \\ & \mathbf{W}_s\mathbf{L}_s & \end{bmatrix} \begin{bmatrix} \mathbf{V}' \\ \mathbf{V}'_s \end{bmatrix} = \begin{bmatrix} \mathbf{W}_L\boldsymbol{\Delta} \\ \mathbf{W}_c\mathbf{V}_c \\ \mathbf{W}_s\boldsymbol{\Delta}_s \end{bmatrix} \tag{7}$$

where $\mathbf{W}_L$, $\mathbf{W}_c$ and $\mathbf{W}_s$ are diagonal weight matrices for surface vertices, control points and skeleton points, respectively.

The whole optimization framework is extended from that of [22]. An initial guess is obtained by solving (7). Then rotation matrices are computed by using SVD, and are plugged into the righthand side of (7) based on the derivative of (3). The two procedures are alternately performed until convergence. This approach computes a natural rotation for each vertex and approximately preserves the edge length and the volume. In addition, the computational complexity of (7) is similar to that of (2) since (7) only appends l rows and columns. Because the skeleton is generally 1D, l is much smaller than the number of surface vertices. Thus the new system does not significantly decrease the efficiency.

Limitation of this framework: Our volume preservation method is seamlessly built upon the ARAP surface modeling framework. Since its carefully designed framework of energy minimization only directly preserves edge length, our model relies on a fundamental assumption to work properly: the skeleton of the model (or its part that we are interested in) can be approximately represented as one straight line. Thus the volume can be divided into sub-volumes, and the sub-volume between two cross sections can be roughly kept by preserving areas of cross sections and lengths of edges on the surface. This is the reason we are using perpendicular rays to generate the cross sections. In practice, many models or their main bodies have this property, especially models from natural objects. Such examples are provided in the experiment section.

2.3. Mesh optimization

Mesh quality is crucial to many mesh editing algorithms, such as ARAP surface modeling. When using cotangent weights, this method is especially sensitive to the mesh quality. Degenerated triangles may result in unstable deformation. Preprocessing the mesh for smoothness can alleviate this problem. However, meshes may be also degenerated during the deformation. Thus it is desirable to perform mesh optimization during deformation, without significantly increasing the computational complexity. This can be done by re-using intermediate results.

Least squares mesh: Least square mesh [23] is an algorithm to improve triangle quality of a surface mesh. The inputs are anchor points and an initial surface mesh. The output is an optimized surface mesh. Using a small subset of m anchor points, a mesh can be reconstructed from connectivity information alone by minimizing the quadratic energy:

$$\|\mathbf{L}_u\mathbf{V}'\|_2^2 + \Sigma_{i=1}^{m}|v'_{a_i} - v_{a_i}|^2 \tag{8}$$

where the v_a are anchor (landmark) points. $\|\mathbf{L}_u\mathbf{V}'\|^2$ tries to smooth the object by minimizing the difference among neighbors, and $\Sigma_{i=1}^{m}|v'_{a_i} - v_{a_i}|^2$ keeps anchor points unchanged. In practice, with m anchors, it is minimized by solving the $(n+m)\times n$ overdetermined linear system:

$$\begin{bmatrix} \mathbf{L}_u \\ \mathbf{I}_a \end{bmatrix} \mathbf{V}' = \begin{bmatrix} \mathbf{0} \\ \mathbf{V}_a \end{bmatrix} \tag{9}$$

The first n rows are the Laplacian constraints, corresponding to $\|\mathbf{L}_u\mathbf{V}'\|^2$, while the last m rows are the positional constraints, corresponding to $\Sigma_{i=1}^{m}|v'_{a_i} - v_{a_i}|^2$. $\mathbf{I}_a$ is the index matrix of $\mathbf{V}_a$, which maps each $\mathbf{V}'_a$ to $\mathbf{V}_a$. The reconstructed shape is generally smooth, with the possible exception of small areas around anchor vertices. The minimization procedure moves each vertex to the centroid of its *1*-ring, since the uniform Laplacian $\mathbf{L}_u$ is used, resulting in good inner fairness.

Detail-preserving optimization: One limitation of the least square mesh is that the shape details and sharp features may be smoothed out. Minimizing $\|\mathbf{L}_u\mathbf{V}'\|^2$ means that each vertex moves toward the centroid of its neighbors (along the red arrow in Fig. 1). This approach can effectively improve the mesh quality but would sacrifice the shape details. Another Laplacian coordinate based approach is called *detail-preserving optimization* [15,14]. The underlying idea is that instead of moving each vertex towards the centroid of its neighbors, it is moved along the tangential direction. This approach removes the difference between the uniform and cotangent weights (red and green arrows in Fig. 1). The linear system to solve now is designed as:

$$\begin{bmatrix} \mathbf{L}_u \\ \mathbf{I}_a \end{bmatrix} \mathbf{V}' = \begin{bmatrix} \boldsymbol{\Delta} \\ \mathbf{V}_a \end{bmatrix} \tag{10}$$

By setting the $\mathbf{L}_u\mathbf{V}' = \boldsymbol{\Delta}$ in the first n rows, where $\boldsymbol{\Delta} = \mathbf{L}_c\mathbf{V}$, the system actually eliminates the difference between $\mathbf{L}_u$ and $\mathbf{L}_c$. In other words, each vertex moves along the tangential plane until the uniform and cotangent weights (red and green arrows in Fig. 1) are colinear. This approach improves the mesh quality while also preserving shape details.

A significant issue is the choice of anchor points. Nealen et al. [14] show that all vertices can be employed as positional constraints using weighted least square. Then the $2n \times n$ linear system to solve is defined as:

$$\begin{bmatrix} \mathbf{W}_L\mathbf{L}_u \\ \mathbf{W}_a\mathbf{I} \end{bmatrix} \mathbf{V}' = \begin{bmatrix} \mathbf{W}_L\boldsymbol{\Delta} \\ \mathbf{W}_a\mathbf{V} \end{bmatrix} \tag{11}$$

where $\mathbf{W}_L$ and $\mathbf{W}_a$ are diagonal matrices defining the weights of the Laplacian coordinates and positional constraints, respectively. This setting does not require any heuristic or user input. The computational complexity of solving system (11) is similar to that of (10), even though the size of the system is almost doubled.

Mesh quality: Since most meshes are represented as or can be converted to sets of triangles, we use the radius ratio [16] to measure the mesh quality. It is defined as:

$$t_i = 2\frac{r}{R} \tag{12}$$

where R and r are the radii of the circumscribed and inscribed circles respectively. The radius ratio takes values within $[0,1]$, and $t_i = 1$ indicates a well shaped triangle. The mean and minimal values of t_i are crucial for the robustness of many deformable models. The goal of mesh optimization is to increase these two values.

2.4. Implementation and acceleration considerations

In each iteration, our system needs to solve two sparse linear systems, (7) and (11), and find the rotation matrix R_i for each vertex. Both linear systems have the similar format $\mathbf{AV}' = b$, where $\mathbf{A}$ consists of the Laplacian matrix $\mathbf{L}$ and indicator matrices. Theoretically, it can be solved analytically by $\mathbf{V}' = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^Tb$, whose computational complexity is $O(n^3)$, where n is the size of $\mathbf{A}$. However, $\mathbf{A}$ is extremely sparse (4–7 nonzero elements in each row) and does not change during deformation. As a result, sparse Cholesky factorization [26] can be used as an initialization. The solution can be rapidly computed by updating b and using forward and backward substitution during interaction. Since the factorized matrices are approximately banded sparse, the computational complexity of solving the two systems is decreased to $O(n)$.

The other computational cost comes from finding the rotation matrix R_i for each vertex with SVD. In this step, since the weights w_{ij} are constant during deformation, they are calculated once and stored. Pre-allocating memory space to all R_i can also improve the efficiency. Another acceleration method is that during the mesh deformation procedure, the previous frame is used as the initial guess instead of solving (7).

When implementing the skeleton-based ARAP method, the skeleton edge information can be stored as a separate data structure, which can be a link list maintaining the connectivity among skeleton points and surface vertices. Thus the surface mesh in our method is exactly the same as the one of other ARAP methods. In the mesh deformation step, the vertices on the surface and the skeleton points are updated together, while in the mesh optimization step, only the surface mesh is considered.

3. Experiments

3.1. Experimental settings

In this section we evaluate our algorithm in terms of volume preserving property and robustness. The volume preserving property comes from the deformation procedure. It is validated in Section 3.2. The robustness benefits from alternately employing mesh deformation and optimization, whose evaluation is in Section 3.3. The C++implementation was run on a Intel Core2 Quad 2.40 GHz CPU with 8G RAM. Fig. 3 shows the seven models used in the experiments. Table 1 displays the statistics of these models, including the number of vertices and faces, the minimal and mean values of the radius ratio.

3.2. Evaluation of volume preservation

Five models are used for evaluation, namely the cylinder, the spiky plane, the cactus, the horse and the armadillo model. To fairly compare our deformation procedure with ARAP surface modeling [22], these models are smoothed and simplified, since ARAP surface modeling may result in corrupted meshes for complex models (Section 3.3).

Without skeleton information, our method works in the same way as ARAP surface modeling. Fig. 4 demonstrates

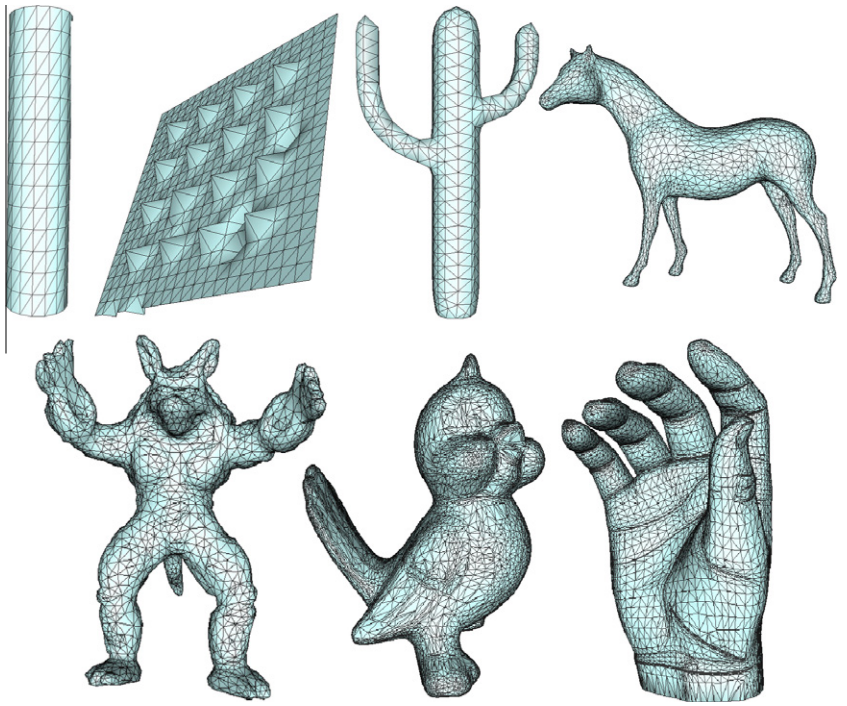

Fig. 3. Models used in our experiments. First row: cylinder, plane with spikes, Cactus, Horse. Second row: Armadillo, Tweety, Hand. Note that the mesh qualities of the tweety model and the hand model are very low.

Table 1
Statistics of models used in experiments, including the number of vertices and faces, the minimal and mean values of radius ratio.

Model	#Vertices	#Faces	T_{min}	T_{mean}
Cylinder	240	448	0.742	0.742
Plane with spikes	441	800	0.104	0.821
Cactus	620	1236	0.377	0.842
Horse	2482	4960	0.036	0.744
Armadillo	2703	5402	0.033	0.835
Tweety	7053	14,102	1.6e − 4	0.665
Hand	7609	15,214	6e − 11	0.602

the rotation recovery and the edge-length preservation. The plane with spikes is a typical scenario. When using linear methods and only allowing translations, the normals of spikes always point to the original direction (Fig. 4c). Multi-resolution can alleviate the problem, but it cannot avoid self-intersections. ARAP surface modeling can find natural rotations for spikes in three iterations (Fig. 4d).

Fig. 5 compares the linear LSE, ARAP surface modeling and our skeleton-based ARAP mesh modeling. The cactus model is a challenging test case due to its long protruding features. With enough iterations, ARAP surface modeling can recover rotations for these features. However, when rotations are large, the bent volume is shrunk. Skeleton constraints can alleviate this problem. The regions marked by black boxes show the volume differences using two methods. Fig. 6 compares ARAP surface modeling with our method on a horse model. Using our method, the volume of the main body is mostly preserved. Fig. 7 shows one more deformation example using an armadillo model.

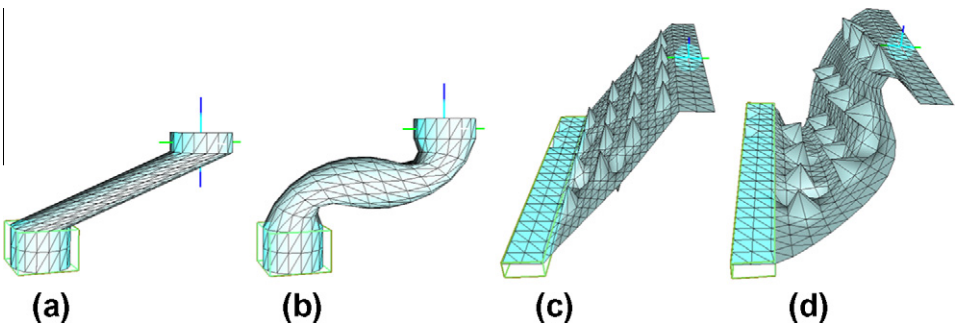

Fig. 4. Cylinder and spiky plane. (a and c) Linear LSE, (b and d) ARAP surface modeling (three iterations). The deformations are achieved by anchoring the bottom and translating the top. Note that only translations are involved.

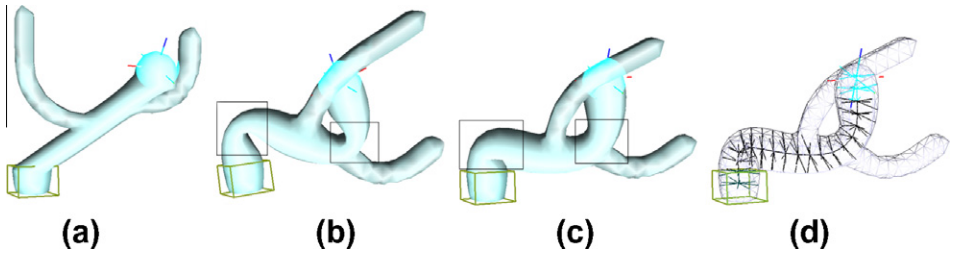

Fig. 5. Dancing cactus. (a) Linear LSE, (b) ARAP surface modeling (100 iterations), (c) using skeleton-based ARAP mesh modeling (100 iterations), (d) skeleton edges and wire frames of (c). The deformations are achieved by anchoring the bottom and translating the top. The black boxes show volume differences between traditional ARAP surface modeling and skeleton based method. Note that only translations are involved.

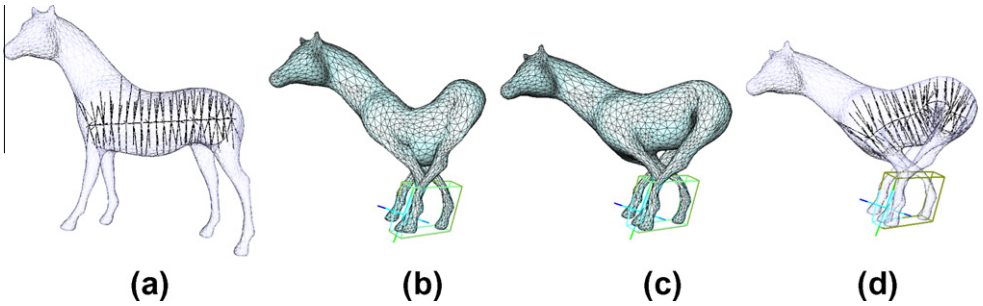

Fig. 6. Running horse. (a) original model with skeleton edges displayed, (b) ARAP surface modeling (three iterations), (c) using skeleton based ARAP mesh modeling (three iterations), (d) skeleton edges and wire frames of (c). The deformations are achieved by anchoring the front legs and translating the rear legs. Only translations are involved.

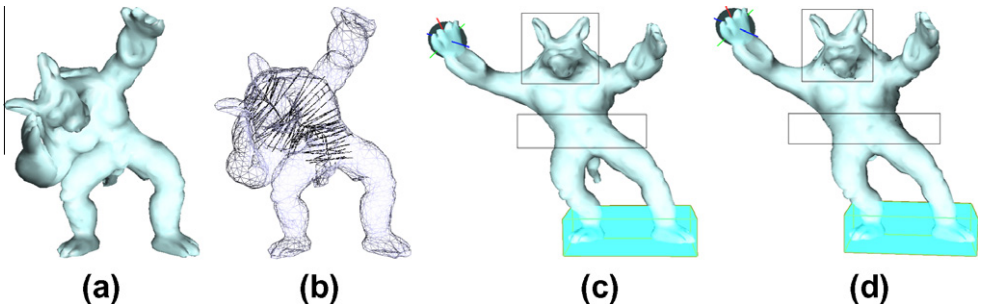

Fig. 7. Armadillo deformation. (a and b) Armadillo shot put. This deformation is obtained by translating and rotating the right arm using our skeleton based method. (b) Displays the skeleton (black lines inside of the body) and the object's wire frame. Volumes are mostly preserved. (c) ARAP surface modeling; (d) skeleton-based ARAP mesh modeling using the same settings as (a) and (b). The deformations are achieved by anchoring the feet and translating the hand. Only translations are involved.

Table 2 shows the relative root mean square errors of edge lengths and volume magnitudes. As expected, the skeleton based method performs much better in terms of volume preservation. It also shows the calculation time for each iteration of different methods. Our method only increases the processing time by about 3% compared to ARAP surface modeling.

Multiple branches: As we discussed in Section 2.2, the skeleton is preferred to be a straight line. However, sometimes the skeleton is complex and has branches. Our model can be extended to handle such multiple-branch cases by applying the above algorithms on each branch and its

Table 2
Errors and processing times of Figs. 5 and 6. RRMS-E stands for relative root mean square errors of edge lengths. RE-V means the relative error of volume magnitudes: abs(original volume − current volume)/(original volume). "Times" is the calculation time (s) for each iteration.

Editing sessions	RRMS-E	RE-V	Times
Fig. 5a	0.126	0.453	0.017
Fig. 5b	0.074	0.131	0.024
Fig. 5c	**0.075**	**0.056**	**0.025**
Fig. 6b	0.068	0.356	0.117
Fig. 6c	**0.040**	**0.125**	**0.121**
Fig. 7c	0.027	0.063	0.132
Fig. 7d	**0.021**	**0.014**	**0.137**

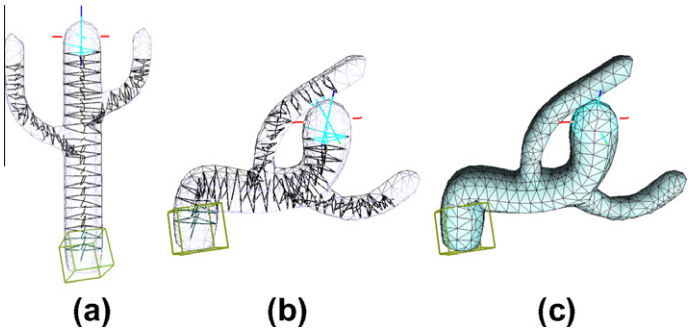

Fig. 8. Cactus deformation. The skeleton has multiple branches and is extracted using the method in [4]. Then our method is applied on each branch. The other settings are the same as Fig. 5.

sub-branches. One more difficulty is how to generate such multiple branches from a surface model. Manually interaction can be tricky and time consuming. We used mesh contraction based skeleton extraction [4] to generate them. We did preliminary experiments on multiple-branch models, such as the Cactus model (Fig. 8). There are five branches or sub-branches in this model, i.e., one in the main body and two in each arm. Compared to the results in Fig. 5, this multiple-branch model also keeps the shape and the volume magnitude of the arms of the Cactus. The relative error of volume magnitude is 0.041 in this case. Fig. 9 shows one more example, i.e., the Armadillo model. It contains more than 10 branches, which are in the main body, the tail, the arms and legs of the Armadillo model. Its relative error of volume magnitude is 0.011.

One limitations of this skeleton-based ARAP mesh modeling method, is that self-intersection may happen (Figs. 5 and 6), so the cross sections may cross, which makes the model less stable. Mechanisms to prevent self-intersection are a valuable addition.

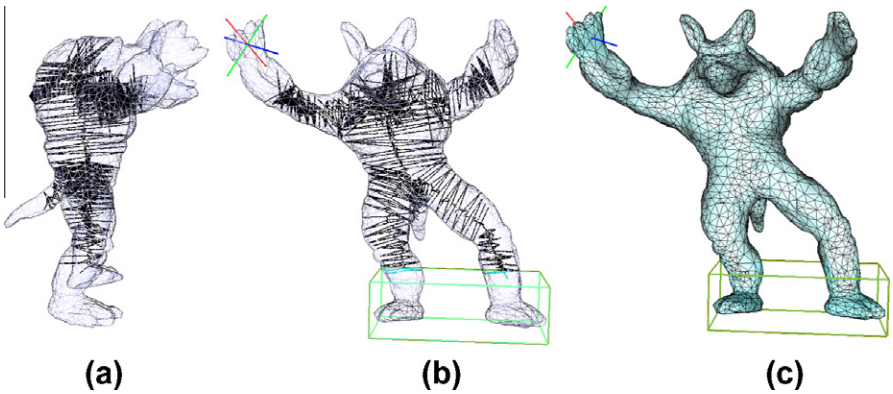

Fig. 9. Armadillo deformation. The skeleton has multiple branches and is extracted using the method in [4]. Then our method is applied on each branch. The other settings are the same as Fig. 7.

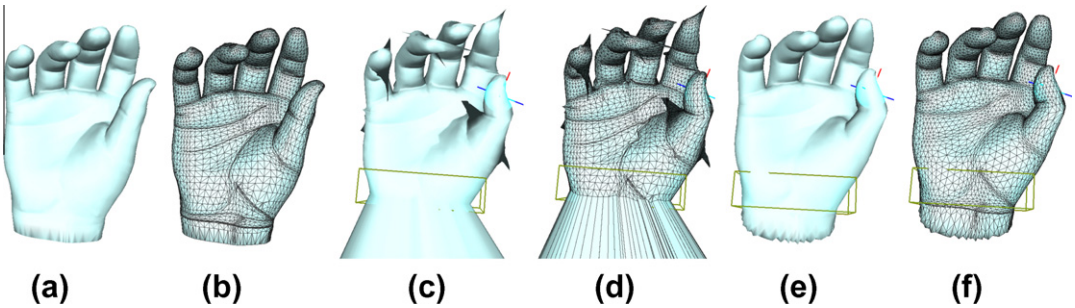

Fig. 10. The thumb of the hand is rotated, and the bottom part (inside of the cuboid) is fixed. (a and b) Original model. (c and d) Result from ARAP surface modeling without mesh optimization. The triangles on the tail and wings are degenerated. (e and f) Result from our method. The deformation is more robust, and the shape details are preserved when introducing mesh optimization.

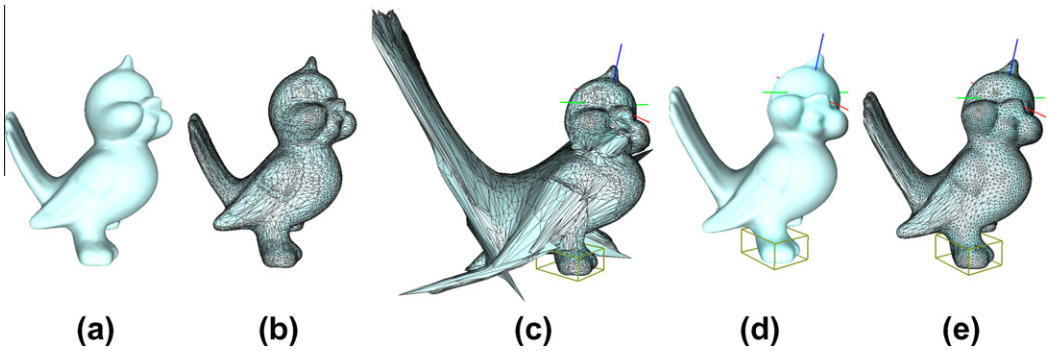

Fig. 11. The head of the tweety is rotated, and the foot is fixed. (a and b) Original model. (c) Result from ARAP surface modeling without mesh optimization. The triangles on the finger and bottom are degenerated. (d and e) Result from our method. The deformation is more robust, and the shape details are preserved when introducing mesh optimization.

3.3. Evaluation of robustness

Two complex meshes are used to evaluate the robustness of our method, the hand and the tweety models. Fig. 10 shows the deformation results of our method and ARAP method without mesh optimization using the hand model, when the thumb of the hand is rotated. Since the model contains many skinny triangles (low value of radius ratio), using classical ARAP surface modeling [22] results in a corrupted mesh. Artifacts can be observed on the tip of the fingers and the bottom of the hand, which all contain many skinny triangles. Since our method employs mesh deformation and optimization alternately in each iteration, the mesh quality is ensured during deformation, making the model more stable. Because detailed preserving mesh optimization is employed, the shape details are kept (Fig. 10a and e). Fig. 11 shows other deformation results using the Tweety model. Its head is rotated and its foot is anchored. Artifacts are observed on the tail and wing, both of which contains many skinny triangles, when using ARAP surface modeling without the optimization procedure. Our method improves the robustness while not affecting the shape details.

Table 3 provides quantitative comparisons between our method and ARAP method without mesh optimization. Since the resulting meshes from the ARAP method are degenerated, the differences of the edge length and volume magnitude compared with the original meshes are very large. Interestingly, the relative root mean square errors of edge lengths (RRMS-E) of our method are also noticeable. The reason is that the mesh optimization affects edge lengths, especially for skinny triangles. These tiny edges can be elongated to improve the mesh quality. Thus the differences compared to the original mesh are increased, i.e., RRMS-E is increased. However, the shape details are preserved, and the volume magnitudes are kept. Thus slightly increased RRMS-E is acceptable here. Our method also produces larger radius ratio because of the benefit of mesh optimization. Although the minimal value of radius ratio is still very small, the overall mesh quality is noticeably improved. One limitation is that our method needs more processing time in each iteration because of the extra optimization step. However, because of the pre-factorization, the computational overhead is only around 10%.

Generally our method is more robust. This additional optimization step in each iteration does not adversely affect the shape details or computational complexity.

Table 3
Comparisons between our method and ARAP method without mesh optimization. RRMS-E, RE-V Times have the same meanings as Table 2. T_{min} and T_{mean} are the same as Table 1.

Editing sessions	RRMS-E	RE-V	Times	T_{min}	T_{mean}
Fig. 10c	>5	>5	0.390	8e −16	0.536
Fig. 10e	**1.178**	**0.041**	**0.431**	**5e − 4**	**0.773**
Fig. 11c	1.143	0.414	0.383	1e −6	0.565
Fig. 11e	**0.736**	**0.047**	**0.409**	**6e − 4**	**0.793**

4. Discussions

We proposed a framework to robustly perform mesh editing. It includes two procedures in each iteration, mesh deformation and optimization, both of which are based on Laplacian coordinates. The deformation procedure uses skeleton information to approximately preserve the volume magnitude without breaking the manifoldness of traditional ARAP or increasing the computational complexity. The optimization procedure improves the mesh quality, which ensures more robust mesh editing. Our method is easy-to-implement and may be useful to systems relying on ARAP techniques.

There are some limitations in our proposed method. First, the skeleton generation is not fully solved. We provided a simple interface to generate one skeleton by user input, and also an automatic approach to obtain multiple branches. However, neither of them is robust enough to handle all models. Since our method focuses on the deformation, we would like to refine the skeleton generation part in the future work. The second limitation is that the skeleton or the branch of the skeleton has to be a straight line. Our volume preserving property is based on the assumption that the volume can be divided into sub-volumes, and the sub-volume between two cross sections can be roughly kept by preserving areas of cross sections and lengths of edges on the surface. Thus it is preferred that the sub-volume is similar to a cylinder. Then

it can be approximated by the straight line (the skeleton or its branch) and perpendicular rays which generate the cross sections.

In the future we would like to incorporate the content-aware deformation [29,11] into our algorithms, to produce more natural results. We are also interested in combining real skeleton-based animation methods. By editing the skeleton, the surface can be deformed automatically. GPU based acceleration is also a promising source of optimization, since most computations come from the matrix manipulations. This method can also be applied to other applications like medical image analysis [32,27].

References

[1] M. Alexa, D. Cohen-Or, D. Levin, As-rigid-as-possible shape interpolation, in: SIGGRAPH '00, 2000, pp. 157–164.
[2] O. Au, C.-L. Tai, L. Liu, H. Fu, Dual Laplacian editing for meshes, IEEE Trans. Vis. Comput. Graph. 12 (3) (2006) 386–395.
[3] O.K.-C. Au, H. Fu, C.-L. Tai, D. Cohen-Or, Handle-aware isolines for scalable shape editing, ACM Trans. Graph. 26 (3) (2007) 83.
[4] O.K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, T.-Y. Lee, Skeleton extraction by mesh contraction, ACM Trans. Graph. 27 (3) (2008) 1–10.
[5] M. Botsch, L. Kobbelt, An intuitive framework for real-time freeform modeling, ACM Trans. Graph. 23 (3) (2004) 630–634.
[6] M. Botsch, M. Pauly, M. Gross, L. Kobbelt, Primo: coupled prisms for intuitive surface modeling, in: SGP '06, 2006, pp. 11–20.
[7] I. Guskov, W. Sweldens, P. Schröder, Multiresolution signal processing for meshes, in: SIGGRAPH '99, 1999, pp. 325–334.
[8] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, H.-Y. Shum, Subspace gradient domain mesh deformation, ACM Trans. Graph. 25 (3) (2006) 1126–1134.
[9] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, H.-Y. Shum, Subspace gradient domain mesh deformation, ACM Trans. Graph. 25 (3) (2006) 1126–1134.
[10] T. Igarashi, T.F.H.J. Moscovich, As-rigid-as-possible shape manipulation, ACM Trans. Graph. 24 (3) (2005) 1134–1141.
[11] V. Kraevoy, A. Sheffer, A. Shamir, D. Cohen-Or, Non-homogeneous resizing of complex models, ACM Trans. Graph. 27 (5) (2008) 1–9.
[12] Y. Lipman, D. Cohen-Or, R. Gal, D. Levin, Volume and shape preservation via moving frame manipulation, ACM Trans. Graph. 26 (1) (2007) 5.
[13] Y. Lipman, O. Sorkine, D. Levin, D. Cohen-Or, Linear rotation-invariant coordinates for meshes, ACM Trans. Graph. 24 (3) (2005) 479–487.
[14] A. Nealen, T. Igarashi, O. Sorkine, M. Alexa, Laplacian mesh optimization, in: GRAPHITE '06, 2006, pp. 381–389.
[15] A. Nealen, O. Sorkine, M. Alexa, D. Cohen-Or, A sketch-based interface for detail-preserving mesh editing, ACM Trans. Graph. 24 (3) (2005) 1142–1147.
[16] P.P. Pébay, T.J. Baker, Analysis of triangle quality measures, Math. Comput. 72 (244) (2003) 1817–1839.
[17] U. Pinkall, K. Polthier, Computing discrete minimal surfaces and their conjugates, Exp. Math. (1993).
[18] S. Schaefer, T. McPhail, J. Warren, Image deformation using moving least squares, ACM Trans. Graph. 25 (3) (2006) 533–540.
[19] X. Shi, K. Zhou, Y. Tong, M. Desbrun, H. Bao, B. Guo, Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics, ACM Trans. Graph. (2007).
[20] W. Song, L. Liu, Stretch-based tetrahedral mesh manipulation, in: GI '07: Proceedings of Graphics Interface 2007, pp. 319–325.
[21] O. Sorkine, Least-squares rigid motion using svd, Technical notes, 2009
[22] O. Sorkine, M. Alexa, As-rigid-as-possible surface modeling, in: SGP '07, 2007, pp. 109–116.
[23] O. Sorkine, D. Cohen-Or, Least-squares meshes, in: SMI '04, 2004, pp. 191–199.
[24] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, H.-P. Seidel, Laplacian surface editing, in: SGP '04, 2004, pp. 175–184.
[25] D. Terzopoulos, J. Platt, A. Barr, K. Fleischer, Elastically deformable models, in: SIGGRAPH '87, 1987, pp. 205–214.
[26] S. Toledo, Taucs: a library of sparse linear solvers. Tel-Aviv University, 2003. <http://www.tau.ac.il/stoledo/taucs>.
[27] X. Wang, T. Chen, S. Zhang, D. Metaxas, L. Axel, Lv motion and strain computation from tmri based on meshless deformable models, in: MICCAI '08, 2008, pp. 636–644.
[28] D. Xu, H. Zhang, Q. Wang, H. Bao, Poisson shape interpolation, Graph. Models 68 (3) (2006) 268–281.
[29] W. Xu, J. Wang, K. Yin, K. Zhou, M. van de Panne, F. Chen, B. Guo, Joint-aware manipulation of deformable models, in: SIGGRAPH '09, 2009, pp. 1–9.
[30] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, H.-Y. Shum, Mesh editing with poisson-based gradient field manipulation, in: SIGGRAPH '04, 2004, pp. 644–651.
[31] R. Zayer, C. Rössl, Z. Karni, H.-P. Seidel, Harmonic guidance for surface deformation. In: EUROGRAPHICS 2005, vol. 24, 2005, pp. 601–609.
[32] S. Zhang, X. Wang, D. Metaxas, T. Chen, L. Axel, Lv surface reconstruction from sparse tmri using Laplacian surface deformation and optimization, in: ISBI, 2009, pp. 698–701.
[33] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, H.-Y. Shum, Large mesh deformation using the volumetric graph Laplacian, ACM Trans. Graph. 24 (3) (2005) 496–503.
[34] D. Zorin, P. Schröder, W. Sweldens, Interactive multiresolution mesh editing, in: SIGGRAPH '97, 1997, pp. 259–268.