Dual Laplacian Editing for Meshes

Oscar Kin-Chung Au, Chiew-Lan Tai, Ligang Liu, and Hongbo Fu

Abstract—Recently, differential information as local intrinsic feature descriptors has been used for mesh editing. Given certain user input as constraints, a deformed mesh is reconstructed by minimizing the changes in the differential information. Since the differential information is encoded in a global coordinate system, it must somehow be transformed to fit the orientations of details in the deformed surface, otherwise distortion will appear. We observe that visually pleasing deformed meshes should preserve both local parameterization and geometry details. We propose to encode these two types of information in the dual mesh domain due to the simplicity of the neighborhood structure of dual mesh vertices. Both sets of information are nondirectional and nonlinearly dependent on the vertex positions. Thus, we present a novel editing framework that iteratively updates both the primal vertex positions and the dual Laplacian coordinates to progressively reduce distortion in parametrization and geometry. Unlike previous related work, our method can produce visually pleasing deformations with simple user interaction, requiring only the handle positions, not local frames at the handles.

Index Terms—Interaction techniques, surface representations, geometric algorithms.

1 Introduction

Mesh editing systems based on differential coordinates (e.g., Laplacian coordinates and gradient field) have been proposed recently [1], [2], [3], [4], [5], [6], [7]. As these editing systems represent local features using differential coordinates defined in a global coordinate system, they face a common transformation problem: The original differential coordinates of a mesh need to be appropriately transformed to accommodate the changes in the local features of the deformed mesh. Existing solutions either use heuristic methods to transform the differential coordinates or require the user to specify the transformations at handles to be propagated. These systems either cannot handle large angle deformation [1], [2], [3] or cannot handle distortion caused by translations of handles [4], [5], [6], [7].

We propose a novel iterative editing framework based on Laplacian coordinates to solve the transformation problem. Our examination of this transformation problem led us to observe that a visually pleasing deformed mesh should have small deviation from the original mesh in both *local parameterization* and *local geometry*. To achieve this, we decompose the global geometry into two sets of scalar data representing local parametrization and local geometry information. The local parametrization information is captured by the coefficients of the Laplace operator defined on the mesh, and the local geometry information is captured by the magnitudes of the Laplacian coordinates. Noting that both sets of information nonlinearly depend on the vertex positions, we believe that the transformation problem cannot be solved satisfactorily using only linear systems as direct solvers. Our framework iteratively updates both the vertex positions and the Laplacian coordinates during editing to minimize distortion in local geometry and parametrization.

In order to avoid tangential drifting during editing, the local geometry should be represented in the directions of surface normals. We first attempted to use the curvature flow Laplace operator since these Laplacian coordinates approximate the curvature normals. However, as each vertex can have an arbitrary number of neighbors and its one-ring neighbors are usually not coplanar, any weighting scheme would produce Laplacian coordinates with tangential components at vertices with extreme connectivity and geometry (e.g., the one-ring structure forms a saddle). This causes an instability problem in the iterative framework. To resolve the instability problem, we propose to edit in the *dual domain* of the input mesh. As the dual vertices of a triangular mesh always have valence three, there is a unique definition of normal and tangent space at each vertex in terms of its one-ring neighbors.

Our framework has several advantages over existing related frameworks. Previous frameworks only considered how to propagate the rotations of the handles to the entire model, but did not address distortion caused by translations of handles [4], [5], [6], [7]. Our system can produce visually pleasing results when the handles undergo large translations and/or large rotations. Our iterative framework automatically updates the nonconstrained vertices to produce a surface with well-oriented global and local details (Fig. 1).

Previous methods require the user to specify the orientations or local frames of handles. Our method can produce naturally deformed shape solely from the positions of handles (being translated or rotated); the user does not need to input the local frames. We provide two choices of user-interfaces: using a region handle if the user wants to directly control the orientation of the handle, or using a point handle if the user wants the system to automatically find the orientation around the handle. The latter interface makes editing process easier since editing 3D models becomes a sequence of simple click-and-drag operations.

Since our system allows users to freely move the handles, the distances between handles may be changed drastically,

- *O.K.-C. Au, C.-L. Tai, and H. Fu are with the Department of Computer Science, Hong Kong University of Science & Technology, Clear Water Bay, Hong Kong. E-mail: {oscarau, fuhb}@ust.hk, taicl@cs.ust.hk.*
- *L. Liu is with the Department of Mathematics, Zhejiang University, Hangzhou, 310027, P.R. China. E-mail: ligangliu@zju.edu.cn.*

Manuscript received 14 July 2005; revised 10 Dec. 2005; accepted 28 Dec. 2005; published online 10 Mar. 2006.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0088-0705.

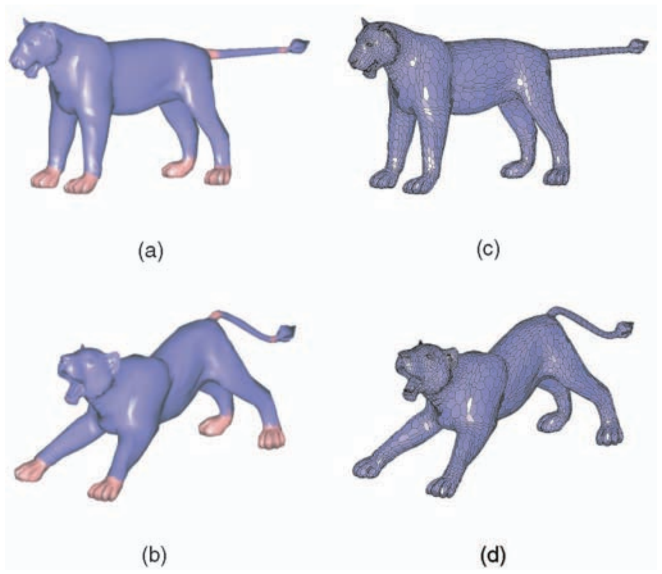


Fig. 1. A deformation example created with our system. (a) Input model. (b) Edited model. (c) and (d) The corresponding dual models. The mesh is edited in the dual domain, but the dual models are hidden from users. Users are only required to specify the handle positions and the local features are automatically rotated smoothly.

causing distortions like anisotropic stretching or squashing. We provide an option to rescale the geometry details in order to reduce such distortions.

The paper is organized as follows: After reviewing related work in Section 2, we present our iterative editing framework based on the curvature flow Laplace operator in Section 3. Section 4 discusses editing in the dual domain and Section 5 presents the option of rescaling the dual Laplacian coordinates. In Section 6, we introduce two new error metrics to measure mesh distortion in the editing results, in terms of parameterization error and geometry error. We describe implementation details and limitations in Section 7 and present the conclusion in Section 8.

2 Related Work

Mesh editing has received extensive attention in the last decade as irregular triangular meshes become a popular representation for 3D models with intricate details. One indispensable property of mesh editing systems is to preserve surface details after editing.

Multiresolution frameworks were first designed for spline surfaces [8] and later applied to subdivision surfaces and irregular meshes [9], [10], [11], [12]. These approaches decompose the mesh into several levels—coarser or smoother versions of the original mesh. The differences between successive levels are defined as local features. They are encoded with respect to the local frames of the lower level mesh. During editing, the details are transformed according to the changes in the local frames of the lower level mesh. Thus, the definition of the local frames and the accuracy of the detail reconstruction are critical in multiresolution editing frameworks.

Recently, differential coordinates have been used as local features for designing mesh editing frameworks [1], [2], [3], [4], [5], [6], [7]. These frameworks allow users to directly specify the positions and/or normal directions of parts of a mesh, called the handles, and the rest of the surface is computed by solving a linear system to minimize shape distortion. Similar to the detail vectors in multiresolution surfaces, the differential coordinates have to be transformed according to the deformed surface in order to produce visually pleasing deformations. However, unlike the multiresolution representations, here there is no lower level surface for defining the local frames for transformation.

Since our editing framework is based on the Laplacian coordinates, we first review Laplacian editing and then describe the limitations of existing differential-based editing frameworks.

2.1 Laplacian Editing

We assume that the input model is a triangular mesh. Let $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n)$ be the mesh vertex positions and i^* be the index set of vertices adjacent to $\mathbf{v}_i$. The *Laplacian coordinate* (LC) of a vertex $\mathbf{v}_i$ is

$$\mathbf{l}_i = \sum_{j \in i^*} w_{ij}(\mathbf{V}_j - \mathbf{V}_i), \tag{1}$$

where w_{ij} is the weight of the edge (i, j) corresponding to the vertex $\mathbf{v}_i$ [13], [14]. Since $\mathbf{l}_i$ is the weighted average difference *vector* of its adjacent vertices to the vertex $\mathbf{v}_i$, it describes the local geometry at $\mathbf{v}_i$. In matrix form, all the LCs can be written as $\mathbf{l} = \mathbf{LV}$, where $\mathbf{L}$ is a $3n \times 3n$ matrix with elements derived from w_{ij}. We refer to $\mathbf{L}$ as the *Laplace operator* and its elements as the *Laplacian coefficients*.

The basic idea of Laplacian editing is to minimize the sum of the squared differences between the LCs before and after editing. The positions $\mathbf{V}'$ of the deformed mesh are found by minimizing $\|\mathbf{LV}' - \mathbf{l}\|^2$, constrained by the positions of some selected vertices as the handles [1], [2]. This is equivalent to solving a sparse linear system

$$\mathbf{AV}' = \mathbf{b} \tag{2}$$

in the least squares sense.

2.2 Transformation Problem

As the original LCs cannot reflect the deformation of the local features (especially the changes of orientation) during editing, reconstructing the mesh from the original LCs with the user-specified constraints would lead to undesired distortion, specifically, shearing and stretching distortion (see Fig. 2b and Fig. 6b).

To avoid distortion, any differential-coordinates-based editing framework must somehow transform the differential coordinates appropriately. This transformation problem is essentially a chicken-and-egg problem: On the one hand, the reconstruction of the deformed surface requires the properly transformed differential coordinates; on the other hand, the transformations of the differential coordinates depend on the unknown deformed mesh.

Several researchers have tried to solve the transformation problem. Lipman et al. [1] used an intermediate reconstructed surface to guess the new orientations of the LCs. Sorkine et al. [2] employed implicitly defined transformations onto the LCs. However, the method is untenable for large angle rotations and anisotropic scalings (i.e., stretching or compressing).

Yu et al. [4] proposed an editing system based on a gradient field, in which the main challenge is also the

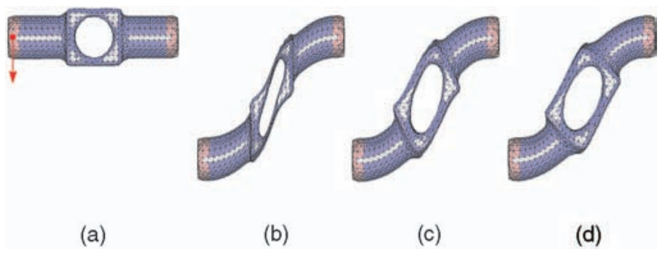


Fig. 2. The iterative process progressively reduces distortion in both local parametrization and local geometry. (a) Input model with handles at two ends. (b) Deformed mesh without reorienting the LCs; triangles are sheared and stretched. (c) After one iteration of our updating method, starting from the mesh in (b). (d) After 10 iterations; the shearing effect is now reduced and the features are deformed naturally.

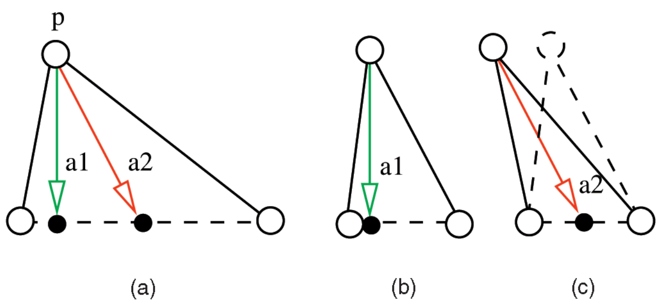


Fig. 3. (a) The local shape at p is represented by either a vector in the normal direction (a_1) or a vector with a tangential component (a_2). (b) Edited surface using the vector in local normal direction. (c) Edited surface using the vector containing a tangential component, thus tangential drift occurs.

transformation problem. Their solution is to explicitly propagate the transformations at the handles to all vertices (weighted by geodesic distances). This idea is extended to perform large mesh deformation using a volumetric graph Laplacian in [7]. Instead of weighted by geodesic distances, Zayer et al. [6] propagated the transformations along harmonic fields. Lipman et al. [5] encoded the vertex differences in local frames and minimize the least squares error of the changes in the local frames. Since these approaches need the orientations (or local frames) of the handles as input, if the handles only undergo translations, there is no change in orientation to be propagated or minimized; thus, these approaches cannot avoid shearing and stretching distortion caused by handle translation (Fig. 2b).

Botsch and Kobbelt [12] built a multiresolution modeling framework using the Laplace equation. They set the LCs (or higher order differential coordinates) to zero, thus only surfaces without any geometry details, like minimal surfaces and thin-plate surfaces, can be built between the handles. Since the differential coordinates vanish, there is no transformation problem to address here. To edit surfaces with local geometry details, they built multiresolution meshes using the technique in [11]. However, for meshes with nonzero genus, the deformed "details-less" surface will contain collapsed and degenerated triangles, making detail encoding in multiresolution framework impractical.

Recently, Sheffer and Krayevoy [15] proposed a representation called pyramid coordinates to encode local features. Like ours, this representation is rotation invariant and, thus, it can avoid the distortion caused by rotation and translation of handles. However, they use an explicit iterative procedure to update the vertex positions, thus any local changes require at least $O(n)$ iterations to propagate to the entire mesh. Their method also involves the computation of the (primal) vertex normals and the tangent plane; thus, it faces the same drifting and convergence problem as our iterative framework that uses the curvature normal (see Section 3).

3 Iterative Updating Framework

We observe that, to preserve local features, the edited mesh should retain two types of information in the original mesh: 1) parameterization information (shapes of triangles); 2) geometry information (sizes of local features). As the Laplace operator can be used to compute planar parameterization for a mesh [16], [17], [18], [19], [20], [21], we consider the Laplacian coefficients as the local parameterization information. Since the LCs are the local differences of vertex positions, they are considered as the local geometry information of the mesh.

To accurately separate the parameterization and geometry, the geometry data should be represented in the normal directions of the local surface. For example, in multiresolution editing frameworks, the detail vectors are either generated with least tangential components [10] or encoded in the normal directions of the lower level (smoother) surface [11]. If the local geometry data contains tangential components, the local features in the edited surface reconstructed from the geometry and parameterization information will contain tangential drifts (Fig. 3). This mostly happens when the vertex sampling is highly nonuniform or the model is stretched or squashed greatly. In Laplacian-based editing, the choice of the weighting scheme w_{ij} is important, since the LCs are a type of local geometry data and thus should be in the local normal directions.

We first choose to use the curvature flow weighting scheme for the computation of the LCs, that is, $w_{ij} = \cot\alpha_i + \cot\beta_i$, where α_i and β_i are the two angles opposite the edge (i, j). The LCs approximate the integrated curvature normals at the vertices [14]: $\mathbf{l}_i = 4Area_i\kappa_i\mathbf{n}_i$, where $Area_i$ is the sum of the areas of the triangles adjacent to the vertex $\mathbf{v}_i$, and $\mathbf{n}_i$ and κ_i are the unit normal and the mean curvature at the vertex $\mathbf{v}_i$, respectively. Note that this weighting scheme is also used for planar conformal parameterization, which preserves local angles [21]. Thus, the curvature flow weighting scheme is suitable for separating the geometry data $(4Area_i\kappa_i)$ and the parameterization data $(w_{ij} = \cot\alpha_i + \cot\beta_i)$.

After separating the vertex positions in global coordinates into the local geometry and parameterization information, we need to address the problem of correctly transforming the LCs. In visually pleasing deformed surfaces, the LC of each vertex should be *normal* to the local surface. Therefore, we aim to keep the LCs in the normal directions and to make the magnitudes of LCs and the Laplacian coefficients similar before and after editing. In general, the computations of the vertex normals and the Laplacian coefficients depend on the vertex positions nonlinearly. Thus, using a single step linear solver to compute the vertex positions cannot give a satisfactory solution. For this reason, we iteratively update both the vertex positions $\mathbf{V}$ and the LCs $\mathbf{l}$ to minimize the

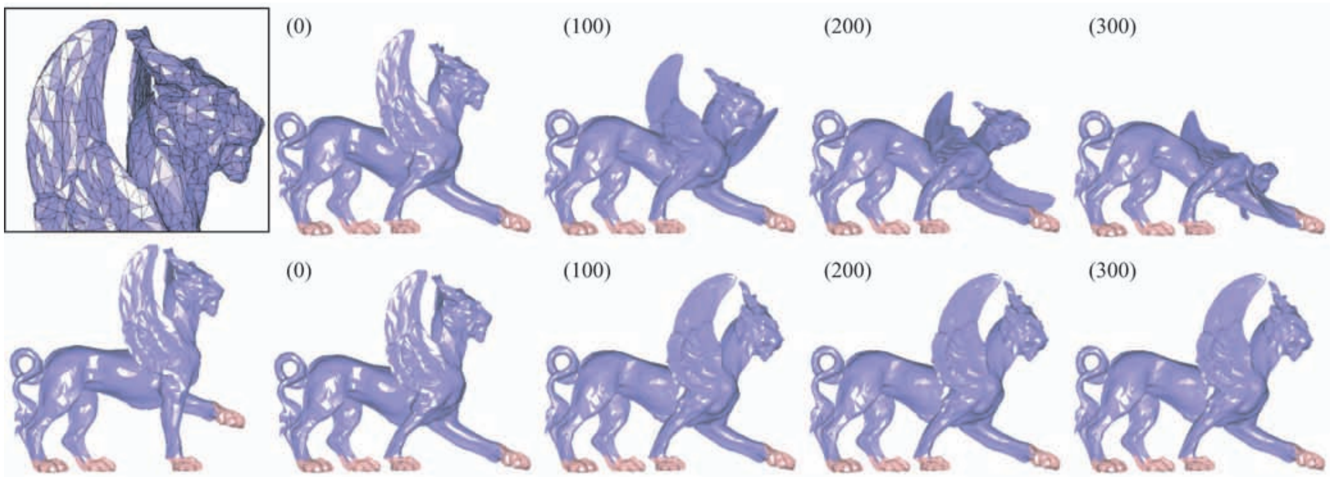


Fig. 4. Editing in the dual domain can improve the stability of our system. The input model has poor sampling quality (leftmost column). The handles at the front legs are moved to new positions (second column). The top row shows the results of our iterative updating algorithm using the LCs in the curvature normal directions. Tangential drifting is accumulated and, thus, the iteration cannot converge. The bottom row is the updating results that use the dual LCs. The normals and the encoding of the dual LCs are well defined. Thus, the updating process is stable and the iteration converges. The number of iterations at each stage is shown in parentheses.

parameterization and geometry distortion. Our framework employs the same linear system given in (2) (Step 1 of the algorithm).

3.1 Algorithm

Let $\mathbf{V}^t$ and $\mathbf{l}^t$ be the vertex positions and the LCs at time t, respectively, such that $\mathbf{V}^0 = \mathbf{V}$ and $\mathbf{l}^0 = \mathbf{l}$. We iteratively update them using the following two steps, until a termination condition (discussed in Section 4) is satisfied:

- **Step 1. Update the vertex positions**. We use the current LCs $\mathbf{l}^t$ to compute the vertex positions $\mathbf{V}^{t+1}$; that is, we solve (2) using the current LCs and the current handle positions as constraints

$$\mathbf{A}^T\mathbf{A}\mathbf{V}^{t+1} = \mathbf{A}^T\mathbf{b}^t, \tag{3}$$

  where $\mathbf{b}^t$ is derived from $\mathbf{l}^t$ and the current handle positions. Note that $\mathbf{A}$ is always defined over the original mesh and is fixed throughout the whole updating process. Thus, this updating step enforces the handle constraints and updates the vertex positions so that the local parametrization of the resulting mesh is similar to the original mesh.
- **Step 2. Update the Laplacian coordinates**. We update the LCs to match the current deformed surface; that is, we compute the new LCs $\mathbf{l}^{t+1}$ from the current vertex positions $\mathbf{V}^{t+1}$. To retain the original feature sizes, we keep the magnitudes of the LCs unchanged. For each vertex $\mathbf{V}_i$, we define its corresponding LC as

$$\mathbf{l}_i^{t+1} = \|\mathbf{l}_i^0\|\mathbf{n}_i^{t+1}, \tag{4}$$

  where $\mathbf{n}_i^{t+1}$ is the unit curvature normal computed from the current vertex positions $\mathbf{V}^{t+1}$ and whose direction may be reversed so as to point inward/outward consistently as the corresponding original curvature normal. The computed unit curvature normals $\hat{\mathbf{n}}_i^{t+1}$ may change between pointing inward or outward (depending on whether the local one-ring structure is convex or concave) during editing. Hence, we use the following definition to ensure that $\hat{\mathbf{n}}_i^{t+1}$ and $\hat{\mathbf{n}}_i^0$ point consistently to the same side of the mesh:

$$\mathbf{n}_i^{t+1} = \begin{cases} \hat{\mathbf{n}}_i^{t+1} & \text{if } (\hat{\mathbf{n}}_i^{t+1}\cdot\check{\mathbf{n}}_i^{t+1})(\hat{\mathbf{n}}_i^0\cdot\check{\mathbf{n}}_i^0) > 0, \\ -\hat{\mathbf{n}}_i^{t+1} & \text{if } (\hat{\mathbf{n}}_i^{t+1}\cdot\check{\mathbf{n}}_i^{t+1})(\hat{\mathbf{n}}_i^0\cdot\check{\mathbf{n}}_i^0) < 0, \\ \check{\mathbf{n}}_i^{t+1}\cdot sign(\hat{\mathbf{n}}_i^0\cdot\check{\mathbf{n}}_i^0) & \text{otherwise,} \end{cases}$$

  where $\check{\mathbf{n}}_i^0$ and $\check{\mathbf{n}}_i^{t+1}$ are the average face normals at vertex $\mathbf{v}_i$ at time 0 and $t+1$, respectively. They are computed as the average of the normals of adjacent triangles at $\mathbf{v}_i$, weighted by the triangle areas. If the original one-ring structure at vertex $\mathbf{v}_i$ is planar, we simply set $\hat{\mathbf{n}}_i^0$ to $\check{\mathbf{n}}_i^0$.

When the iteration converges, the Laplace operators (using the curvature flow weighting) $\mathbf{L}^{t+1}$ defined over the new vertex positions and $\mathbf{L}^0(=\mathbf{L})$ tend to be similar, as do their coefficients, thus they retain the original parameterization information. The LCs are well oriented (in the curvature normal directions) and their magnitudes are also maintained, making the local features similar to those of the original mesh. Thus, the iterations minimize parametrization distortion while keeping local features similar to the original ones. Fig. 2 shows several intermediate results of this iterative process.

This iterative framework, however, has some drawbacks. When the input mesh has poor sampling quality (containing slim triangles) or complex geometry, the iterative updating process may fail to converge (Fig. 4, top row). In general, the one-ring neighbors are not coplanar and the LCs may have tangential components, regardless of which weighting scheme or vertex normal definition is used. This is because there is no common normal direction for all planes formed by the one-ring neighbors; any local encoding will contain tangential component in some of these planes. This causes tangential drifts in the iterative updating process. Moreover, since the curvature normal and the average face normal at a given vertex are not parallel to each other, the inward/outward determination of the curvature normal is not reliable. In addition, due to the irregular connectivity and complex normal computation, it is difficult to determine the convergence conditions. In the next section, we propose to perform iterative editing using dual LCs, which can eliminate all these problems, thanks to the regular and simple connectivity of dual meshes.

As the methods in [1] and [2] use the uniform Laplace operator, the LCs may have large tangential component. However, since their examples are either well sampled or are edited with small changes in the distances between the handles, the drifting effect is not obvious. In order to provide a more general editing framework that can accept nonuniformly sampled models and large handle movements, we need to address the issue of encoding local geometry in the normal direction.

4 Dual Laplacian Editing

In this section, we will show that editing in the dual domain can eliminate the instability in our iterative framework. First, we will describe the construction of dual mesh and the definition of dual LC. Then, we present our improved iterative updating algorithm and some editing examples.

4.1 Dual Mesh and Dual Laplacian Coordinates

We construct the dual mesh consisting of vertices positioned at the centroids of the faces of a primal (original) mesh. There is a one-to-one mapping between the primal vertices (faces) and the dual faces (vertices). The conversion from the primal vertex positions $\mathbf{V}$ to the dual vertex positions $\tilde{\mathbf{V}} = (\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \dots, \tilde{\mathbf{v}}_{n_d})$ can be described by a full-rank matrix $\mathbf{D}$ ($\tilde{\mathbf{V}} = \mathbf{DV}$), where $\mathbf{D}$ is constructed from the vertex-face incident matrix (each row is repeated three times for the x, y, z coordinates) and is normalized such that the sum of each row is equal to one [22].

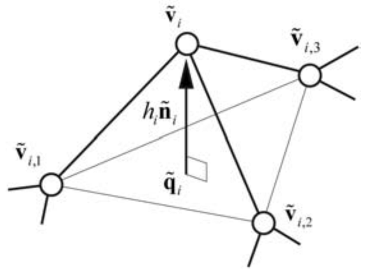
We assume that the input meshes to be edited are triangular meshes. Under this assumption, one important property of the dual meshes is that the valence of every dual vertex is equal to three. This fixed one-ring structure provides a simple and unique representation for each dual vertex $\tilde{\mathbf{v}}_i$, in terms of the plane defined by its one-ring neighbors $\tilde{\mathbf{v}}_{i,j}$, $j = \{1, 2, 3\}$:

$$\tilde{\mathbf{v}}_i = \tilde{\mathbf{q}}_i + h_i \tilde{\mathbf{n}}_i = \sum_{j \in \{1,2,3\}} \tilde{w}_{i,j} \tilde{\mathbf{v}}_{i,j} + h_i \tilde{\mathbf{n}}_i, \tag{5}$$

where $\tilde{\mathbf{q}}_i$ is in the plane that contains the base triangle $\triangle \tilde{\mathbf{v}}_{i,1} \tilde{\mathbf{v}}_{i,2} \tilde{\mathbf{v}}_{i,3}$, and $\tilde{\mathbf{n}}_i$ is the *outward* plane normal (Fig. 5). Thus, $\tilde{w}_{i,j}$ are the barycentric coordinates of $\tilde{\mathbf{q}}_i$ corresponding to the base triangle and their sum is one. h_i is the signed magnitude (depending on whether the one-ring structure is convex or concave) of the normal component. Note that the encoding $(\tilde{w}_{i,1}, \tilde{w}_{i,2}, h_i)$ is unique and rotation invariant.

By rearranging the terms in (5), we define the *dual Laplacian coordinate* (dual LC) of a dual vertex to be the normal component in (5) and derive the Laplacian coefficients from $\tilde{w}_{i,j}$:

$$\tilde{\mathbf{l}}_i = -h_i \tilde{\mathbf{n}}_i = \sum_{j \in \{1,2,3\}} \tilde{w}_{i,j} (\tilde{\mathbf{v}}_{i,j} - \tilde{\mathbf{v}}_i), \tag{6}$$



$\tilde{\mathbf{v}}_i$
$\tilde{\mathbf{v}}_{i,3}$
$h_i\tilde{\mathbf{n}}_i$
$\tilde{\mathbf{v}}_{i,1}$
$\tilde{\mathbf{q}}_i$
$\tilde{\mathbf{v}}_{i,2}$

Fig. 5. Illustration of the one-ring structure of dual vertices.

or, in matrix form,

$$\tilde{\mathbf{l}} = \tilde{\mathbf{L}}\tilde{\mathbf{V}} = \tilde{\mathbf{L}}\mathbf{DV}. \tag{7}$$

Now, the dual LC is exactly in the normal direction of the base triangle formed by the neighbors of $\tilde{\mathbf{v}}_i$. Thus, there will be no tangential drifts in the reconstruction. Since the orientation of the base triangle normal is known (we always use the outward normal), updating the dual LCs during the iteration is easier and more stable than using the curvature normals in the primal domain.

Since matrix $\mathbf{D}$ is full rank and matrix $\tilde{\mathbf{L}}$ has rank $n_d - 1$, given the original dual LCs $\tilde{\mathbf{l}}$ of a given mesh and one fixed primal vertex position, one can compute the rest of the primal vertex positions by solving the normal equation of (7) without error. As in the original Laplacian editing, if there is more than one constrained primal vertex, the deformed (primal) mesh can be found by minimization in the least squares sense:

$$\arg\min_{\mathbf{V}'} \left\| \tilde{\mathbf{L}}\mathbf{D}\mathbf{V}' - \tilde{\mathbf{l}} \right\|^2. \tag{8}$$

This is equivalent to solving a sparse linear system $\tilde{\mathbf{A}}\mathbf{V}' = \tilde{b}$ in the least squares sense. Thus, $\mathbf{V}'$ can be solved from the normal equations $\tilde{\mathbf{A}}^T\tilde{\mathbf{A}}\mathbf{V}' = \tilde{\mathbf{A}}^T\tilde{b}$.

4.2 Algorithm of Dual Laplacian Editing

Let $\tilde{\mathbf{V}}^t$ and $\tilde{\mathbf{l}}^t$ be the dual vertex positions and the dual LCs at time t, respectively, such that $\tilde{\mathbf{V}}^0 = \tilde{\mathbf{V}}$ and $\tilde{\mathbf{l}}^0 = \tilde{\mathbf{l}}$. We iteratively update them using the following two steps, until a termination condition is satisfied:

- **Step 1. Update the dual vertex positions**. To enforce the current constraints from the handles in the primal domain, we first compute the primal vertex positions $\mathbf{V}^{t+1}$ using the current dual LCs $\tilde{\mathbf{l}}^t$; that is, we solve the following equations

$$\tilde{\mathbf{A}}^T\tilde{\mathbf{A}}\mathbf{V}^{t+1} = \tilde{\mathbf{A}}^T\tilde{b}^t, \tag{9}$$

  where $\tilde{b}^t$ is derived from $\tilde{\mathbf{l}}^t$ and the current (primal) handle positions. The new dual vertex positions are then computed by $\tilde{\mathbf{V}}^{t+1} = \mathbf{DV}^{t+1}$.
- **Step 2. Update the dual Laplacian coordinates**. We update the dual LCs to match the current deformed surface; that is, we compute the new dual LCs $\tilde{\mathbf{l}}^{t+1}$ from the dual vertex positions $\tilde{\mathbf{V}}^{t+1}$. For each dual vertex $\tilde{\mathbf{v}}_i$, its corresponding dual LC is defined as

$$\tilde{\mathbf{l}}_i^{t+1} = -d_i^{t+1}\tilde{\mathbf{n}}_i^{t+1}, \tag{10}$$

  where $\tilde{\mathbf{n}}_i^{t+1}$ is the outward normal defined over the base triangle $\triangle \tilde{\mathbf{v}}_{i,1}^{t+1} \tilde{\mathbf{v}}_{i,2}^{t+1} \tilde{\mathbf{v}}_{i,3}^{t+1}$ and d_i^{t+1} is a scaling value. Note that, now, the normals always point outward and are well defined and, hence, the determination of their inward/outward orientations is no longer required unlike the case of curvature normal. To retain the original feature sizes, we keep the magnitudes of the dual LCs unchanged: $d_i^{t+1} = h_i$. Alternatively, by changing the scaling values, we can also iteratively update the scales of the dual LCs and, hence, also update the geometry. The rescaling of the dual LCs will be discussed in the next section.

We terminate the iteration when the maximum ratio of the changes in vertex positions between two successive time steps is less than a given threshold. Once again, when the iteration converges, the Laplace operators (using the barycentric coordinates) $\tilde{\mathbf{L}}^{t+1}$ defined over the new dual vertex positions and $\tilde{\mathbf{L}}^0 (= \tilde{\mathbf{L}})$ tend to be similar, thus retaining the original parameterization. The magnitudes and local directions of the LCs are also maintained, making the local features similar to the original ones. Therefore, this new iterative updating framework minimizes the local geometry and parametrization distortion.

4.3 Discussion

Note that the updating of the dual LCs is nonlinear when h_i is nonzero (Step 2 of our algorithm). We can think of the degree of nonlinearity at a vertex as the ratio of the magnitude of h_i to the area of the base triangle. The convergence speed of the iteration depends on the nonlinearity of the local geometry. The vertex positions require more iterations to converge if the magnitudes of h_i's are relatively large. Note that if all h_i's are zero, the reconstructed surface is a minimal surface (no geometry information) with the given handle positions as boundary constraints. In this case, the process will terminate after a single updating iteration since the system is totally linear.

Our iterative method in dual domain is basically a nonlinear Gauss-Seidel method that solves for the dual vertex positions $\tilde{\mathbf{V}}$ and the dual LCs $\tilde{\mathbf{l}}$ iteratively. To determine under what conditions the iteration will converge, we first consider the following explicit iterative framework without handle constraints:

$$\tilde{\mathbf{v}}_i^{t+1} = \sum_{j \in \{1,2,3\}} \tilde{w}_{i,j} \tilde{\mathbf{v}}_i^t + \tilde{\mathbf{l}}_i^t, \tag{11}$$

$$\tilde{\mathbf{l}}_i^{t+1} = -h_i \tilde{\mathbf{n}}_i^{t+1}. \tag{12}$$

The updating process is stable if the absolute row sum of its Jacobian matrix is less than 1, which depends on the values of h_i and $\tilde{w}_{i,j}$ and the base domain sizes. The above explicit updating system ((11) and (12)) will converge if $max_{j \in \{1,2,3\}} |\tilde{w}_{i,j}| + |6 h_i e_i / Area_i| < 1$, where e_i and $Area_i$ are the maximum edge length and the area of the base triangle, respectively.

Our editing framework essentially replaces the updating rule (11) with an implicit updating method (and include handle constraints). This will not affect the robustness and, in fact, will increase the convergence speed. Our system does not directly check the above convergence condition; instead, it decreases the updating step size (initially it is 1) if the current updating step increases the error in the linear system (in order to decrease the absolute row sum in the updating). In practice, we have not encountered any situations in which the iteration failed to converge. In all our editing experiments, the solver converges very fast and is stable even when the input models have poor triangle quality (see the bottom row of Fig. 4) or when the handles are moved rapidly. Note that when the iteration converges, all the LCs are normal to their corresponding base triangles, so we only consider the error in the vertex positions instead of including the error in the directions of LCs.

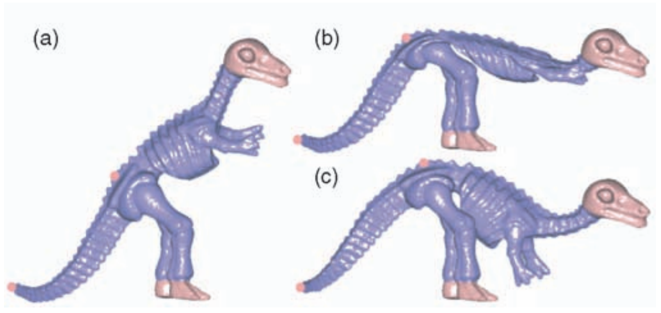


Fig. 6. Editing with our framework. (a) Input model, with handles at the head, feet, back and tail end. (b) Translating the head without reorienting the dual LCs. (c) Same as (b), but with the dual LCs reoriented. Shearing occurs in (b), while in (c) the local features of the body are rotated and deformed smoothly.

Fig. 6 shows a deformation example involving only translations of handles. It is noteworthy that our framework produces well-oriented local features even when the handles are only translated, not rotated. Previous methods like [4], [5], [6], [7] generate results similar to Fig. 6b because there is no change of handle orientations to be propagated or minimized as required by these methods. Other methods that implicitly solve for the local transformations [2], [3] would give better results, similar to the deformed model obtained after one iteration of our framework (as in Fig. 2c). However, the distortion would become obvious under large handle translations because a desired deformed model would have large local rotations at the free vertices and the implicit methods used in [2], [3] can only handle small rotation angles well.

Fig. 7 illustrates our simple point-handle interface; the local orientation at the point handle is automatically decided by our system. Our system does not require local frames as input. Note that since a point handle contains only one vertex, it is impossible to determine the local orientation of the handle directly (since at least three vertices are required to determine a rigid transformation). By updating the orientations of the LCs of the free vertices and point handles, our system can iteratively improve the deformation. This provides a simple click-and-drag editing interface, which is easy to use, especially for novice users. Fig. 8 shows the editing of a model with fine details; both point handles and region handles are used in this example. In all the examples (except Fig. 1), we only show the edited models in the primal domain, as the dual meshes are hidden from the user.

By default, we use the original dual LCs and the original vertex positions as the initial values in the iteration because they are the most natural choices and provide a visually smooth transition to the deformed mesh during interactive

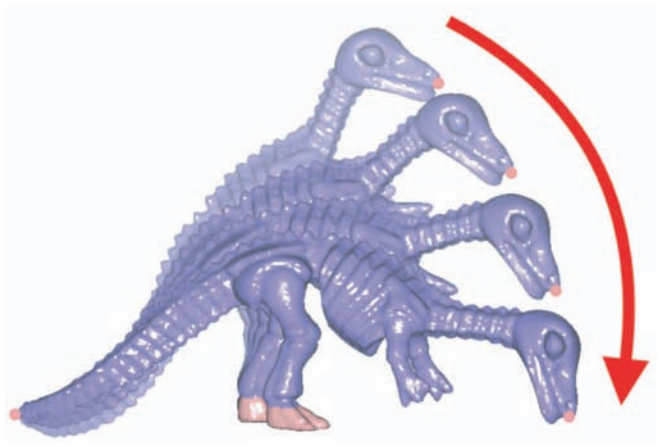


Fig. 7. Editing by only translating point handles. The point handles are at the tail end and nose tip. The nose tip is moved by the user and the body is oriented automatically by our system.

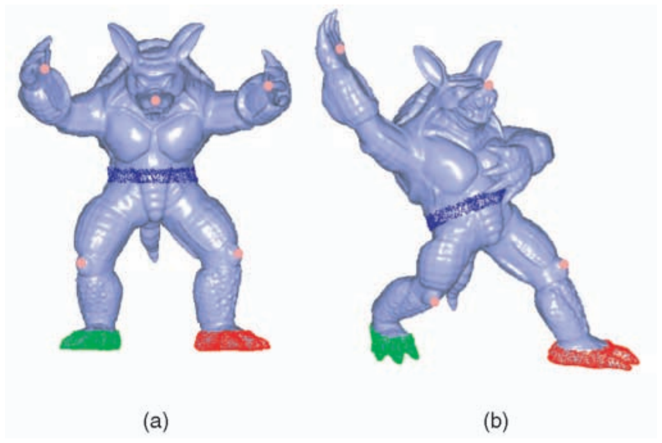


Fig. 8. An editing example of armadillo model. (a) Input model, with three region handles and five point handles. (b) Edited model.

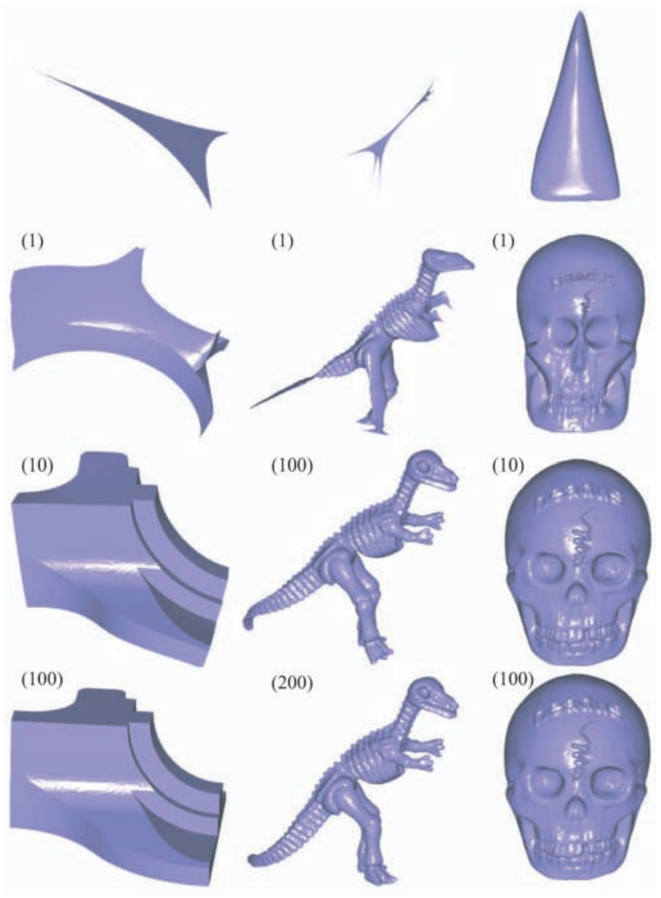


Fig. 9. Reconstruction without using the directions of the original dual LCs; only the scales of the dual LCs and the Laplacian coefficients are used. A few vertices (three to six) are selected as handles. The top row shows the minimal surfaces reconstructed when the zero vector is used as the initial values. The remaining rows are the reconstruction results of iteratively updating the meshes in the first row. The number of iterations is shown in the parentheses.

editing. However, the converged result is independent of the initial vector field (directions of the dual LCs) and are fully determined by the scales of the LCs and the Laplacian coefficients. Fig. 9 demonstrates that our system can reconstruct a model from only the original parameterization and geometry information, without using the directions of the original dual LCs. In these examples, the zero vector is used as the initial values of the iterative process (thus, produces minimal surfaces). All the models converge to the original shapes after 10 to 100 iterations, depending on the geometry complexity. The error evaluation details are given in Section 6.

Similar to our method, the framework of Sheffer and Kraevoy [15] updates the vertex positions iteratively during reconstruction, but their updating method is an explicit procedure. In our method, we update the vertex positions implicitly to enforce a linear relation between the dual LCs and the vertex positions and update the dual LCs explicitly based on the nonlinear relation. The error caused by the change in each LC (Step 2 of each iteration) is distributed evenly to the entire surface in one iteration (by minimizing the error in the least squares sense). Hence, our system always requires fewer iterations than a fully explicit system like the method in [15].

5 Rescaling of Dual LCs

When the user's manipulation of a handle causes drastic changes of the distances between handles, stretching or squashing distortion occurs. In such editing situations, edge lengths are modified drastically, thus the angles between adjacent faces are also changed greatly. In these cases, merely reorienting the dual LCs, while keeping their magnitudes unchanged, cannot produce deformation with small parameterization distortion (Fig. 10b and Fig. 11b). Rescaling the Laplacian coordinates in order to maintain the angles between adjacent triangles, thus modifying the feature sizes, can produce a more natural result with less parameterization distortion (Fig. 10c and Fig. 11c).

Since the dual LCs are linear combinations of the vertex positions (as well as the dual vertex positions), they have the same scaling factors as the vertex positions under isotropic scaling. Based on this fact, we let the user have the option to rescale the dual LCs to be of magnitude equal to the average edge lengths to reduce anisotropic scaling. We use the square root of the base triangle area as the average edge length and reset the scaling factor in (10):

$$d_i^{t+1} = h_i\sqrt{Area_i^{t+1}/Area_i^0}, \tag{13}$$

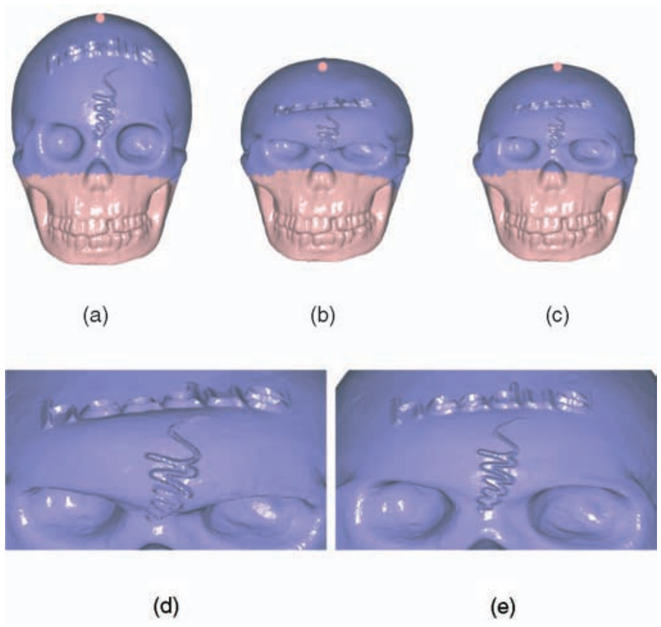


Fig. 10. Rescaling of the dual LCs improves the reconstruction of local features. (a) Original model, with a point handle and a region handle. Editing the mesh by moving the top handle downward, (b) without rescaling the LCs and (c) with rescaling the LCs. Notice that the undesired distortion depends on the local geometry complexity.

where $Area_i^0$ and $Area_i^{t+1}$ are the base triangle areas in the dual meshes at time 0 and $t+1$, respectively.

Fig. 10 shows an editing example in which distortion occurs if the magnitudes of the dual LCs are kept the same as the original dual LCs (Fig. 10b). It is observed that undesired distortion occurs nonuniformly over the surface, depending on the local geometry complexity. In such cases, it is difficult to design a scaling field for the dual LCs, as required by previous methods [3], [4], [7]. With our rescaling option, the system automatically eliminates most of the distortion and obtains a more natural deformation (Fig. 10c). Fig. 11 shows another deformation example demonstrating the effect of rescaling the LCs. The handles are moved closer to each other, so the space between them becomes smaller and cannot accommodate the big global features of the original body. By rescaling the LCs, the features are automatically scaled, giving a better visual result. Note that anisotropic stretching is sometime desired (Fig. 12); hence, the rescaling is provided only as an option.

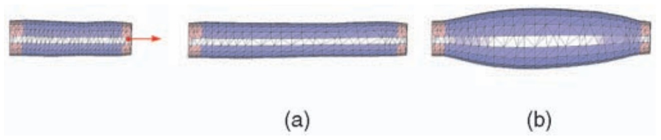


Fig. 12. (a) Distortion in local parameterization due to stretching. (b) Rescaling the dual LCs to reduce anisotropic stretching, but the deformed shape may not be what the user wants.

6 Error Evaluation

Our deformation algorithm is based on the idea of separating the parameterization and geometry information of the mesh. Therefore, it is natural to measure the mesh difference before and after the deformation in terms of the differences in the Laplacian coefficients (the parameterization error E_p) and the differences in the magnitudes of the dual LCs (the geometry error E_g). We define the error metrics as:

$$E_p = \sqrt{\frac{1}{n_d}\sum_i \sum_{j\in\{1,2,3\}} (w_{i,j}^0 - w_{i,j}^t)^2},$$

$$E_g = \sqrt{\sum_i \left(h_i^0 - h_i^t\right)^2},$$

where $w_{i,j}^0$ and $w_{i,j}^t$ (h_i^0 and h_i^t) denote the weights (normal components) computed from the original dual mesh and the deformed dual mesh, respectively. To facilitate comparison of the geometry error among models of different sizes, we first scale the input models to fit within a unit boundary box before editing and error evaluation.

Table 1 gives the error estimation of the deformation examples in this paper. The errors are measured for the meshes obtained when the iteration converges. In all our deformation experiments, we found that the reorientation and rescaling of the dual LCs greatly reduced the parameterization error. In addition, we observed that, with the reorientation of the LCs, the geometry error is similar to (or smaller than) the deformation without the reorientation. With the rescaling of the dual LCs, the geometry error may be slightly increased, but it is evenly distributed over the surface and the features are scaled smoothly and, hence, it further reduces parameterization distortion. This shows that both our reorientation and rescaling methods can

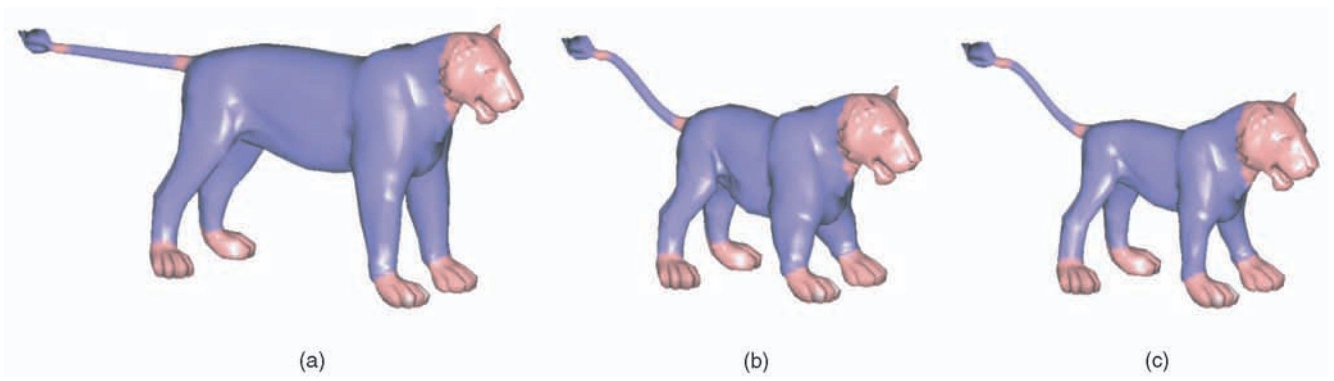


Fig. 11. A baby lion cloned from her mother. (a) Input model. (b) Deformed model with reorientation of the dual LCs, but without rescaling them. (c) Deformed model with reorientation and rescaling of the dual LCs.

TABLE 1
Errors (E_p/E_g) of the Deformation Examples Shown in this Paper

No updating	With reorientation	With reorientation and rescaling
(2b) 0.055/0.0059	(2d) 0.015/0.0049	-
(6b) 0.029/0.0087	(6c) 0.0040/0.0017	-
(7) 0.042/0.0075	(7) 0.0018/0.00082	-
-	(10b) 0.0065/0.0038	(10c) 0.0041/0.014
-	(11b) 0.0084/0.0028	(11c) 0.0074/0.0078

improve the parameterization distortion while retaining the geometry details well.

Table 2 lists the error estimation of the reconstruction examples shown in Fig. 9. As the iteration progresses, all the examples converge to the original shapes and both the parameterization and geometry errors tend to zero.

7 Implementation Details and Limitations

All the examples presented in this paper were created on a 2.0GHz Pentium IV computer with 512MB memory. The most time-consuming part of our algorithm is solving the sparse linear system. We use the direct solver in [23] in our implementation. The factorization of the normal equation may take a longer time, but it is precomputed only once (when the user finishes demarcating the handles). At each iteration step, only back-substitutions are performed to solve the system. The running time is the same for each iteration since the updating procedure is fixed. In all our experiments, it usually takes 10-20 iterations for the deformation to converge. The number of iterations depends on the geometry complexity of the model, and it is independent of the total number of vertices and the speed of the handle movement. Therefore, interactive rate can be achieved in all the applications. Table 3 shows the time required for the factorization and back-substitution for the examples in this paper.

Our editing framework has several limitations. First, like other mesh editing frameworks based on differential coordinates, the input mesh has to be a 2-manifold. If the mesh contains open boundaries, the boundary vertices have to be the handles (as the boundary conditions of the linear system). Also, if the connectivity is changed or a different set of constrained vertices are selected, the linear system has to be rebuilt.

TABLE 2
Errors (E_p/E_g) of the Reconstruction Examples Shown in Fig. 9

Iteration	Fandisk	Dinosaur	Skull
0	0.95/0.099	0.82/0.057	0.020/0.059
1	0.46/0.076	0.18/0.034	0.053/0.020
10	0.0038/0.0015	0.032/0.0062	0.00042/8.30e-5
100	3.20e-5/2.34e-5	0.0010/0.00062	1.03e-13/9.47e-14
200	-	3.80e-4/2.45e-4	-

TABLE 3
Time Required for Factorization and Back-Substitution (in Seconds)

Model	Number of vertices	Factorization	Back-substitution
Lion	5000	0.360	0.016
Feline	4176	0.28	0.016
Dinosaur	14000	1.63	0.047
Skull	20002	3.38	0.078
Armadillo	60000	31.7	0.43

Our system does not support editing with rotation angles between successive handles larger than π. This is because one of our objectives is to keep the user interface simple: Only the final handle positions are required as user input. There are infinite possible transformations to reach the final position of a handle. Our system always chooses the transformation with the smallest rotation angle ($< \pi$), since a larger angle would give greater distortion. To perform editing with a rotation angle larger than π, extra user input is required. A possible approach [4] is, given the (user specified) transformation for each handle, the system can interpolate the transformations to the remaining vertices. Alternatively, the user can specify more handles such that the rotation angles between successive handles are smaller than π. For the purpose of a simple and unified interface, we prefer the latter solution. Fig. 13 shows two editing examples of a bar model using large translation (Fig. 13a, two handles) and large rotations of handles (Fig. 13b, four handles).

8 Conclusions and Future Work

We present an iterative framework to solve the transformation problem. The mesh parameterization information is captured by the coefficients of the Laplacian operator, and the local geometry information is represented by the LCs. Our framework minimizes the local parameterization and geometry distortion in deformation, including the distortion caused by handle translation. Point handles are supported to provide a simple interface such that there is no need for the user to specify the orientations of the handles. To have a stable framework for deforming meshes with poor sampling quality and complex geometry, we propose to perform the deformation and editing in the dual mesh domain. We also provide an option for updating the scales of the LCs to modify the feature sizes. Two new error metrics are introduced to measure mesh distortion.

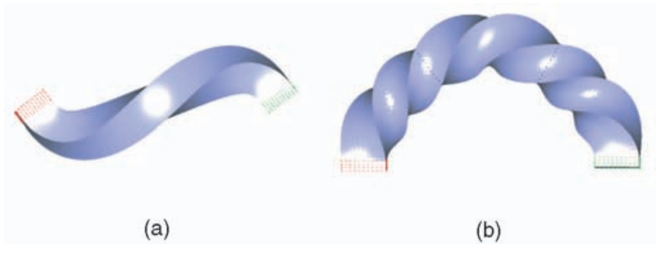


Fig. 13. Editing examples involving large translation (left) and large angle rotations (right).

Transforming a mesh geometry to rotation invariant information is an interesting idea. Working on such information is often easier than working directly on directional vector fields. Potential future work includes exploring such scalar representations to develop new algorithms for signal processing, compression, and remeshing.

Acknowledgments

The authors thank the anonymous reviewers for their valuable comments which helped improve the exposition of the paper. This work received support from the Hong Kong Research Grant Council (Project No. HKUST6295/04E), National Natural Science Foundation of China (Grant No. 60333010, 60503067), and Zhejiang Provincial Natural Science Foundation of China (Grant No. Y105159).

References

[1] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel, "Differential Coordinates for Interactive Mesh Editing," *Proc. Conf. Shape Modeling Int'l,* pp. 181-190, 2004.
[2] O. Sorkine, Y. Lipman, D. Cohen-Or, M. Alexa, C. Rössl, and H.-P. Seidel, "Laplacian Surface Editing," *Proc. Symp. Geometry Processing,* pp. 179-188, 2004.
[3] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or, "A Sketch-Based Interface for Detail-Preserving Mesh Editing," *ACM Trans. Graphics,* vol. 24, no. 3, 2005.
[4] 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. Graphics,* vol. 23, no. 3, pp. 644-651, 2004.
[5] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or, "Linear Rotation-Invariant Coordinates for Meshes," *ACM Trans. Graphics,* vol. 24, 2005.
[6] R. Zayer, C. Rössl, Z. Karni, and H.-P. Seidel, "Harmonic Guidance for Surface Deformation," *Computer Graphics Forum, Proc. Eurographics,* 2005.
[7] 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. Graphics,* vol. 24, no. 3, 2005.
[8] D. Forsey and R. Bartels, "Hierarchical B-Spline Refinement," *Proc. ACM SIGGRAPH '88,* pp. 205-212, 1988.
[9] D. Zorin, P. Schroder, and W. Sweldens, "Interactive Multiresolution Mesh Editing," *Proc. ACM SIGGRAPH '97,* pp. 259-268, 1997.
[10] I. Guskov, W. Sweldens, and P. Schröder, "Multiresolution Signal Processing for Meshes," *Proc. ACM SIGGRAPH '99,* pp. 325-334, 1999.
[11] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel, "Interactive Multi-Resolution Modeling on Arbitrary Meshes," *Proc. ACM SIGGRAPH '98,* pp. 105-114, 1998.
[12] M. Botsch and L. Kobbelt, "An Intuitive Framework for Real-Time Freeform Modeling," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 630-634, 2004.
[13] G. Taubin, "A Signal Processing Approach to Fair Surface Design," *Proc. ACM SIGGRAPH '95,* pp. 351-358, 1995.
[14] M. Desbrun, M. Meyer, P. Schröder, and A.H. Barr, "Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow," *Proc. ACM SIGGRAPH '99,* pp. 317-324, 1999.
[15] A. Sheffer and V. Krayevoy, "Pyramid Coordinates for Morphing and Deformation," *3D Data Processing, Visualization, and Transmission,* pp. 68-75, 2004.
[16] M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic Parameterizations of Surface Meshes," pp. 209-218, 2002.
[17] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes," *Proc. ACM SIGGRAPH,* pp. 173-182, 1995.
[18] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least Squares Conformal Maps for Automatic Texture Atlas Generation," *ACM Trans. Graphics,* vol. 21, no. 3, pp. 362-371, 2002.
[19] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle, "Conformal Surface Parameterization for Texture Mapping," *IEEE Trans. Visualization and Computer Graphics,* vol. 6, no. 2, pp. 181-189, Apr.-June 2000.
[20] X. Gu and S.-T. Yau, "Global Conformal Surface Parameterization," *Proc. Symp. Geometry Processing,* pp. 127-137, 2003.
[21] U. Pinkall and K. Polthier, "Computing Discrete Minimal Surfaces and Their Conjugates," *Experimental Math.,* vol. 2, no. 1, pp. 15-36, 1993.
[22] G. Taubin, "Dual Mesh Resampling," *Proc. Conf. Pacific Graphics,* pp. 94-113, 2001.
[23] S. Toledo, "Taucs: A Library of Sparse Linear Solvers, Version 2.2," Tel-Aviv Univ., http://www.tau.ac.il/stoledo/taucs/, 2003.

Oscar Kin-Chung Au is a PhD student at the Hong Kong University of Science and Technology, from which he received the BSc and MPhil degrees in computer science in 2001 and 2003, respectively. His research interests include computer graphics and polygonal mesh modeling and processing.

Chiew-Lan Tai received the BSc degree in mathematics from the University of Malaya, the MSc degree in computer and information sciences from the National University of Singapore, and the DSc degree in information science from the University of Tokyo. She is an associate professor of computer science at the Hong Kong University of Science and Technology. Her research interests include geometric modeling, computer graphics, and reconstruction from architecture drawings.

Ligang Liu received the BS degree in mathematics in 1996 from Zhejiang University, China. In 2001, he received the PhD degree in mathematics, also from Zhejiang University. From 2001 to 2004, he was an associate researcher at the Internet Graphics Group, Microsoft Research Asia. Since 2004, he has been an associate professor in the Department of Mathematics at Zhejiang University. His current research interests include geometric modeling and processing, interactive computer graphics, and image processing.

Hongbo Fu received the BS degree in information science from Peking University, China, in 2002. He is a PhD candidate in computer science at the Hong Kong University of Science and Technology. His primary research interests fall in the field of computer graphics with an emphasis on interactive surface editing techniques.