3D Research manuscript No.
(will be inserted by the editor)

Luoting Fu · Levent Burak Kara · Kenji Shimada

Feature, Design Intention and Constraint Preservation for Direct Modeling of 3D Freeform Surfaces

Received: date / Accepted: date

Abstract Direct modeling has recently emerged as a suitable approach for 3D free-form shape modeling in industrial design. Its has several advantages over the conventional, parametric modeling techniques, including the natural user interactions, as well as the automatic feature-preserving shape deformation algorithms under the hood. However, current direct modeling packages still lack several capabilities critical for product design, such as managing aesthetic design intentions, and enforcing dimensional, geometric constraints.

In this paper, we describe a novel 3D surface editing system capable of jointly accommodating aesthetic design intentions expressed in the form of surface painting and color-coded annotations, as well as engineering constraints expressed as dimensions. The proposed system is built upon differential coordinates and

Grant information of the article will be placed here in the final version.

Luoting Fu

Visual Design and Engineering Lab, Department of Mechanical Engineering, Carnegie Mellon University

E-mail: luoting.fu@cmu.edu

Levent Burak Kara *(Corresponding author)*

Visual Design and Engineering Lab, Department of Mechanical Engineering, Carnegie Mellon University

Tel.: +1-412-268-2508

E-mail: lkara@cmu.edu

Kenji Shimada

Department of Mechanical Engineering, Carnegie Mellon University

Tel.: +1-412-268-3614

E-mail: shimada@cmu.edu

constrained least squares, and is intended for conceptual design that involves frequent shape tuning and explorations. We also provide an extensive review of the state-of-the-art direct modeling approaches for 3D mesh-based, freeform surfaces, with an emphasis on the two broad categories of shape deformation algorithms developed in the relevant field of computer graphics.

Keywords Shape editing · Freeform geometry · Feature-preserving · Design Intention · Geometric constraints

1 Introduction

Direct modeling has recently emerged as a popular approach for 3D free-form shape modeling in industrial design. It differs from the conventional, parametric modeling techniques in two major ways. Firstly, direct modeling provides a natural editing interface that enables the designers to pick any geometric entity or region, drag or rotate it in desirable directions, and obtain the deformed shape. Secondly, direct modeling provides shape deformation algorithms that ensure the edited shape to respond smoothly and intuitively to the manipulations applied by the user, thus providing an editing modality analogous to virtual sculpting. In comparison, parametric modeling does not offer such intuitiveness.

Because of the above advantages, direct modeling seems promising for product design involving frequent editing of freeform shapes. Several CAD venders, such as SolidWorks and Autodesk, have included free-form direct modeling features in their packages. Existing direct modeling packages, however, lack several features critical for product design. Specifically, we have identified two main issues below.

Firstly, the process of product design entails constant changes to the shape being edited. Or, as the Industrial Design Society of America has put it, “Design is the process of taking something from its existing state and moving it to a preferred state”. During this exploratory process, the designers discover shape features of higher aesthetic values, and navigate away from features that are less visually pleasing. Intuitively, one feature that would greatly facilitate the shape space explorations is to accommodate different aesthetic design intentions for different regions of the shape. For instance, the designer should be able to specify that “I like this region. Keep it visually unchanged while I manipulate and deform the adjacent regions,” or “This portion is almost ok now, but the adjacent portion needs more bending. I wish that the former has a higher stiffness than the latter, hence less deformation, in response to my subsequent manipulations.”

Secondly, the process of product design is a synergetic effort of both aesthetics and engineering. Even though product designs first conceptualize as artistic representations such as freehand sketches or clay models, they are eventually evolved into engineering models, with engineering requirements such as geometric dimensions, packaging size and internal volumes added. Intuitively, one feature that would facilitate the negotiation of aesthetics and engineering requirements is to consider engineering constraints early in product design, and constrain the space of aesthetic surfaces with them. For instance, the designer should be able to specify that "The distance between this feature and the other one should be fixed at 10 centimeters, no matter how I deform the regions surrounding them."

Existing direct modeling CAD packages, such as Inventor Fusion by Autodesk (Autodesk 2012), fall short in both aspects. The geometric constraint solving system and the freeform direct modeling system are mutually exclusive: if a constrained part is edited using direct modeling, the constraints will be no longer binding.

Meanwhile, the field of computer graphics has contributed several model deformation algorithms that seem promising for direct modeling. Such algorithms are able to support the designer by automatically preserving ascetically significant features, such as the curvature distribution over a 3D surface. However, in their existing form, they are still not fully suitable for direct modeling. The main barriers, detailed later in Section 2, are rooted in their intended use for computer graphics and especially character animations. With rare exceptions, they treat every region of the geometry equally as a feature, and attempt to preserve them uniformly in a least squares sense. As a result, the designers, if using those systems for industrial design, will not have a fine-grained control to prescribe different deformation behaviors for different regions. Worse yet, because of the least squares formulation, engineering features such as pair-wise distances and shape rigidity are approximately, rather than exactly, preserved. While sufficient for graphics and animations, their utility for industrial design is limited.

To address those issues, we describe a feature preserving direct modeling system built upon differential coordinates (Alexa 2003), Laplacian surface editing (Sorkine et al 2004) and constrained least squares (Golub and Van Loan 1996; Björck 1996). Using our approach, aesthetic design intentions, such as different levels of rigidity to deformations, are expressed in the form of surface painting and color-coded annotations, and applied to different regions of the shape. The annotations are then converted into a smoothly diffused scalar field over the surface, similar to non-homogeneous material stiffness, and affect how much deformation each region exhibits. Engineering constraints are expressed as geometric dimensions, and converted to linear or quadratic constraint equations that augment the least squares formulation of surface deformation. This way, aesthetic intentions and engineering constraints are preserved during shape deformation.

1.1 Contribution and significance

Our technical contribution is a novel 3D surface editing system capable of jointly accommodating aesthetic design intentions expressed in the form of surface painting and color-coded annotations, as well as engineering constraints expressed as dimensions.

As previously discussed, this system is intended to facilitate early conceptualization in industrial design that heavily involves product shape tuning and the explorations of many alternatives. The preservation of design intentions narrows down the space of the shapes resulting from deformations, and assists the designer in the shape exploration by eliminating the manual preservation of desired aesthetic features. The preservation of dimensional constraints introduces engineering considerations early in conceptual design to facilities its negotiation with aesthetics.

The capabilities of our system are well aligned with the roles that freeform surfaces play in industrial design. As Sederberg and Parry (1986) have observed, freeform surfaces can be categorized into aesthetic surfaces featuring particular visual appearances, functional surfaces satisfying the engineering requirements, and surfaces serving as smooth transitions between two other surfaces. Obviously, the aesthetic surfaces and transitional surfaces fulfill different design intentions, and they each require specialized treatments during deformation. Aesthetic surfaces should be subject to different levels of feature preservation: some are strict, allowing rigid transformations only, while some can be relaxed, permitting slight changes of local curvatures. Transitional surfaces should be subject to even less feature preservation, exhibiting large deformations when adjacent aesthetic surfaces are repositioned. Moreover, the functional surfaces necessitate the preservation of engineering constraints during deformation.

Apart from product design, the proposed approach could also be useful for the end-user customization of a product. In such cases, the aesthetic design intentions and engineering dimensions from the designer constrain the space of the deformed shapes to one that is pre-determined by the designer. As a result, the customized shapes that the end-users generate via direct manipulation would still be subject to the influence of the designer, and share the similar aesthetic features as well as the same engineering constraints with the reference design.

1.2 Outline of this paper

The reminder of this paper is organized as follows. Section 2 reviews existing feature-preserving shape deformation algorithms. Two broad categories of algorithms, surface deformation (Section 2.1) and space deformation (Section 2.2) are compared and discussed in relation to the proposed approach. Section 3 analyzes the problem of preserving design intents and engineering constraints in direct modeling, and presents our solution. Finally, Section 4 showcases three design examples using our approach.

2 Feature-preserving shape deformation algorithms

Computer graphics is one of the fields that has a long-standing interests in techniques that deform a shape per the user's direct manipulations, yet preserve the characteristic visual features of the shape. This interest in feature-preservation arises primarily in character animation, where a continuous sequence of a character's action is obtained by deforming a key frame or a static template.

A widely accepted design principle of feature-preserving modeling systems is that the shape being deformed should only undergo rigid or conformal transformations, including rotation, translation and isotropic scaling, while other forms of affine transformations such as anisotropic scaling and shearing should be minimized (Sorkine and Alexa 2007; Lipman et al 2008; Cohen-Or 2009). In theory, if a shape is merely rotated, translated and isotropically scaled, it is perceptually similar as before, as shown in Figure 1. In practice, it is sometimes undesirable or impossible to rigidly transform the entire shape, for example, when an animator wishes to wave the arms of a character while fixing its torso. It is then up to each different algorithm to distribute the deformations to different parts of the model, so as to optimize the objective functions based on the principle of "maximal rigid transformations and minimal shearing." Two broad categories of algorithms, both reviewed below, have been developed to this end.

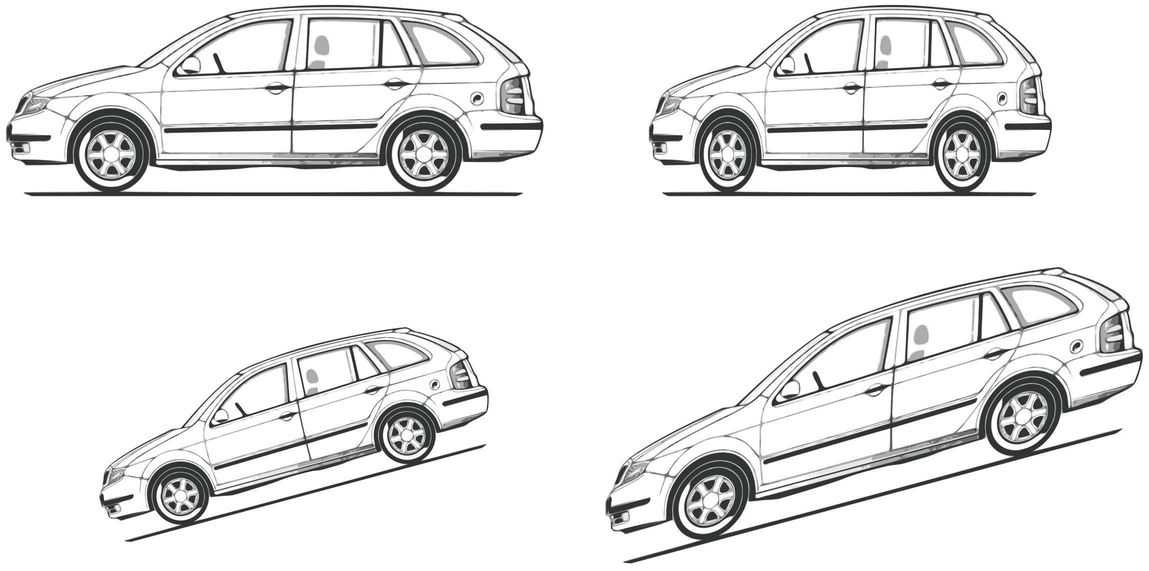


Fig. 1: The various transformations of the side view of a car. Top-left: the original drawing. Bottom-left: after 15° rotation and $\times 0.75$ uniform scaling, the resulting shape looks similar to the original. Top-right: A $\times 0.75$ scaling in the horizontal direction produces a shape that is visually different. Bottom-right: A vertical shearing of 15° also produces a shape that is visually different.

2.1 Surface deformation

Feature-preserving surface deformation algorithms are generally based on the explicit preservation, before and after deformation, of a representation called the differential coordinates. In the following sections, we first review, in detail, a representative approach known as Laplacian surface editing (Sorkine et al 2004) that is based on Laplacian coordinates. We show the definitions of the Laplacian coordinates, and describe the preservation of Laplacian coordinates when the model is subject to user-guided deformations. We then review other related surface editing algorithms.

2.1.1 Laplacian coordinates

A popular form of differential coordinates for mesh-based models is the Laplacian coordinates (Alexa 2003; Sorkine et al 2004; Lipman et al 2005a; Botsch and Sorkine 2008). To show the definition of the Laplacian coordinates, we first set up the notations as follows. Let $\mathscr{M} := (V,T)$ be a non-manifold, triangular surface mesh representing the model being edited. V denotes the set of vertices, where $|V| = n_v$ is the number of vertices, and $\mathbf{v}_i = (v_i^x, v_i^y, v_i^z)$ is the Cartesian coordinates for vertex i, $i \in V$. T denotes the triangular faces, from which the 1-ring neighborhood operator $N(i)$ of vertex i can be derived as the set $N(i) := \{j|(i,j) \in T\}$. $d_i := |N(i)|$ is then the degree or valence of vertex i.

The Laplacian coordinates $\delta_i \in \mathbb{R}^3$ of vertex i is then defined as

$$\delta_i = L_{ii}\mathbf{v}_i + \sum_{j \in N(i)} L_{ij}\mathbf{v}_j,$$

or, equivalently in the vector-matrix product form

$$\delta = \underbrace{\begin{pmatrix} L & & \\ & L & \\ & & L \end{pmatrix}}_{:=L_D} \mathbf{v}, \tag{1}$$

where $\delta \in \mathbb{R}^{3n_v}$ and $\mathbf{v} \in \mathbb{R}^{3n_v}$ are vectors obtained by concatenating the components δ_i and $\mathbf{v}_i$ column-wise, respectively. For instance, $\mathbf{v} = (v_1^x, v_2^x, \ldots, v_{n_v}^x, v_1^y, v_2^y, \ldots, v_{n_v}^z)^{\mathrm{T}}$. $L \in \mathbb{R}^{n_v \times n_v}$, the repeated diagonal block of L_D, is the Laplacian matrix, also known as the discrete Laplace operator, computed from the topology and the geometry of $\mathscr{M}$.

There are several different formulations for L. The simplest formulations, the graph Laplacian (Sorkine et al 2004; Lipman et al 2005a) stemming from graph theory, is purely topological:

$$L_{ij} := \begin{cases} -d_i & i = j \\ 1 & j \in N(i) \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Alternatively, there exists a more computationally-involved formulation (Pinkall and Polthier 1993; Meyer et al 2002b), defined as,

$$L_{ij} := \begin{cases} -\frac{1}{a_i} & i = j \\ \frac{1}{2a_i}(\cot\alpha_{ij} + \cot\beta_{ij}) & j \in N(i) \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where the angles α_{ij} and β_{ij} are defined in Figure 2. a_i denotes the modified Voronoi area of vertex i (Meyer et al 2002b). To ensure that a_i is contained in $N(i)$, the boundaries of the Voronoi area in obtuse triangles are defined differently from those in non-obtuse triangles, as follows:

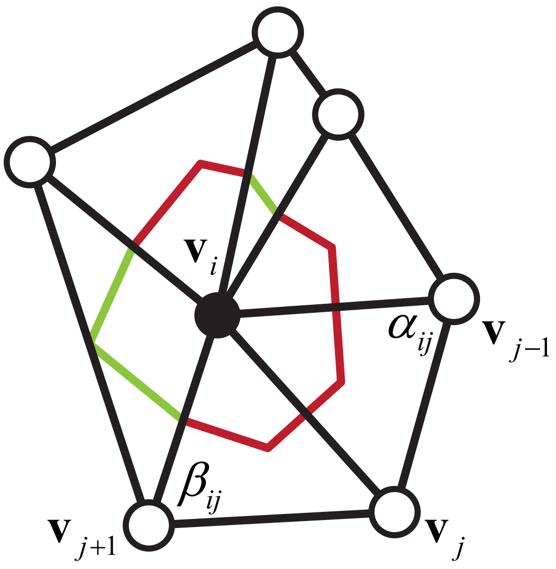


Fig. 2: The 1-ring neighborhood of vertex i. The red and green line segments define the boundaries of the modified Voronoi area of vertex i. The red boundaries, occurring only in non-obtuse triangles, are the perpendicular bisectors of the the edges incident to vertex i. The green boundaries, occurring only in obtuse triangles, connect the midpoints of the edges of the triangle. If the angle at vertex i is non-obtuse or obtuse, two or three midpoints are connected, respectively.

- In non-obtuse triangles, the boundaries are conventionally defined as the intersecting perpendicular bisectors of the the edges incident to vertex i.
- In obtuse triangles, if the angle at vertex i is obtuse, then the boundaries are alternatively defined as the two line segments connecting the midpoint of the opposing of edge vertex i and the edges incident to vertex i.
- Otherwise, the boundary is defined as the single line segment connecting the two mid-points of the edges incident to vertex i.

The first case in shown in red, and the latter two cases are shown in green in Figure 2. As a result, a_i leads to a perfect tiling to the mesh surface, meaning that $\sum_{i\in V} a_i$ equals the surface area of $\mathscr{M}$.

Compared to the graph Laplacian defined in Eq. (2), the cotangent Laplacian in Eq. (3) has been noted to provide more favorable numerical qualities, more accurate approximations of the geometric characteristics of the mesh, and less sensitivity to mesh irregularities (Botsch and Sorkine 2008). In fact, Meyer et al (2002b)

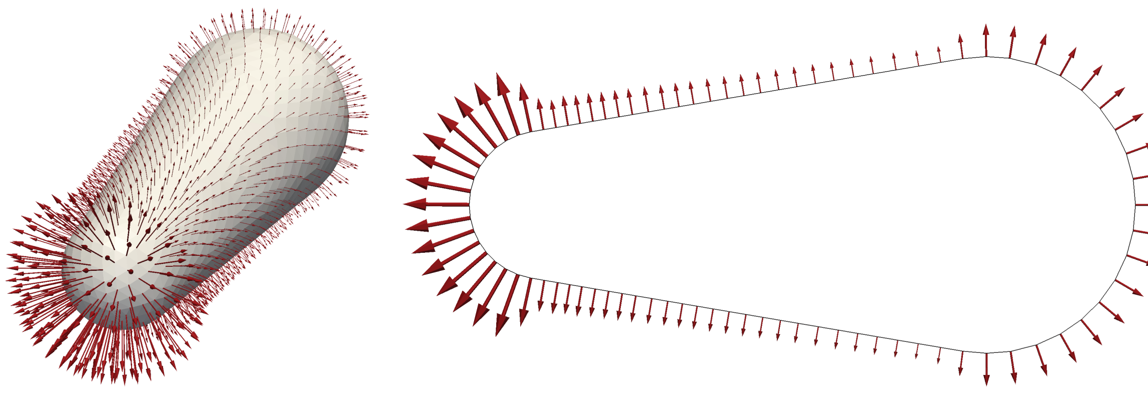


Fig. 3: The Laplacian coordinates, drawn as red arrows, over a shape composed of two spheres and a truncated cone. Left: the Laplacians of all vertices viewed in 3D. Right: the Laplacians of vertices on a 2D cross-section through the axis.

and Xu (2004) have established that Eq. (3) converges to the Laplace-Beltrami operator and provides an approximation of the mean curvature normal, a characteristic property significantly related to the visual and aesthetic features of the shape.

Including the above-discussed convergence and approximation properties, L has the following properties.

Convergence L converges to the Laplace-Beltrami operator in the limit of mesh refinement (Meyer et al 2002b; Xu 2004). As a result, the solution $\mathbf{s} \in \mathbb{R}^{n_v}$ of the linear system

$$L\mathbf{s} = 0 \qquad \text{subject to} \qquad \forall i \in B, s_i = b_i, \tag{4}$$

where B and b_i are the given Dirichlet boundary conditions, is a smoothly diffused scalar field over $\mathscr{M}$. Eq. (4) is known as the Laplace equation and $\mathbf{s}$ is known as the harmonic field (Zayer et al 2005). An engineering instance of such equations is the steady-state temperature distribution over the vertices of $\mathscr{M}$ subject to constant temperature at the boundary.

Approximation δ_i approximates the mean curvature normal at vertex i (Meyer et al 2002b; Sorkine et al 2004). Namely, the orientation of δ_i is aligned with the normal at vertex i, and the magnitude, *i.e.*$\| \delta_i \|_2$, equals the mean curvature at vertex i. Intuitively, the Cartesian coordinates $\mathbf{v}_i$ is a vector emanating from the origin to vertex i, whereas the Laplacian coordinates δ_i is a vector anchored at vertex i, pointing to its normal, as is shown in Figure 3. A significant implication of this comparison is that, if $\mathbf{v}_i$ is rotated, it is rotated about the origin in the global frame; if δ_i is rotated, it is rotated about vertex i in a local frame defined by its 1-ring neighborhood.

Sparsity L is a sparse matrix containing a small number of non-zero elements, with density $\rho = (\sum_{i\in V} d_i + n_v)/n_v^2$. For a mesh with $n_v = 1\times 10^4$ and an average degree of 6, the density of L is on the level of 1×10^{-4}.

Translation invariance Let H_t be a transformation consisting of translations only. Then

$$\delta = L_D\mathbf{v} = L_D H_t(\mathbf{v}).$$

Commutativity w.r.t.rotation Let H_r be a transformation consisting of rotations only. Then

$$H_r(\delta) = H_r\big(L_D(\mathbf{v})\big) = L_D H_r(\mathbf{v}).$$

Decomposition $L = D^{-1}L_{sym}$, where the diagonal matrix $D = \text{diag}(a_1, a_2, \cdots, a_{n_v})$ and L_{sym} is symmetric, with

$$(L_{sym})_{ij} = \begin{cases} -1 & i = j \\ \frac{1}{2}(\cot\alpha_{ij} + \cot\beta_{ij}) & j \in N(i) \\ 0 & \text{otherwise.} \end{cases}$$

With this decomposition, efficient Cholesky decomposition (*e.g.* Chen et al 2008) can be used in lieu of LU decomposition (*e.g.* Duff 2004; Davis 2004), when solving linear systems involving L, such as Eq. (4).

Rank deficiency Rank$(L) < n_v$, which follows from the observation that the columns of L are linearly dependent, *i.e.*, $\forall i \in V, \sum_{j\in V} L_{ij} = 0$ (Sorkine 2006). Mathematically, the linear system of Eq. (1), with known L, δ and unknown $\mathbf{v}$, has non-unique solutions. Intuitively, given a per-vertex curvature distribution, infinitely many meshes can be reconstructed from the prescribed curvatures. Any two of the feasible solutions are related by a rigid body translation. To obtain a unique solution, at least one vertex has to be anchored in space.

Other formulations for the Laplacian matrix exist. (*e.g.* Floater 2003). The trade-offs involved are studied by Grinspun et al (2006) and Wardetzky et al (2007). The cotangent formulation in Eq. (3) is recommended for shape modeling according to Botsch and Sorkine (2008).

All the above-mentioned formulations are defined for triangular meshes only. Alexa and Wardetzky (2011) have recently extended the cotangent formula to the general case of polygonal meshes.

2.1.2 Laplacian surface editing

The previous section reviews the definitions and properties of the differential coordinates δ and the Laplacian matrix L of the mesh $\mathscr{M}$. This section now proceeds to describe the preservation of those coordinates, when $\mathscr{M}$ is deformed.

We begin by setting up the problem of direct manipulation based shape modeling, also known as direct modeling. The given inputs of this problem include, first of all, the model $\mathscr{M} = (V,T)$ itself. Then, to capture the user's manipulations such as locking some vertices of $\mathscr{M}$ while dragging some other vertices, define $C_A \subset V$ as the set of *anchors* which the user specifically wishes to be fixed during deformation. Define $C_H \subset V$ as the set of *handles* which the user specifically wishes to be moved to the target positions $\mathbf{u}' \in \mathbb{R}^{3|H|}$. Together, $C_A \cup C_H$ are the set of constrained vertices, and $C_A \cap C_H = \varnothing$.

The goal is to compute the deformed mesh $\mathscr{M}'$, specifically the deformed vertex coordinates $\mathbf{v}'$, such that the following general conditions are met:

$$\forall j \in C_A, \mathbf{v}'_j = \mathbf{v}_j, \tag{5a}$$

$$\forall k \in C_H, \mathbf{v}'_k = \mathbf{u}'_k, \tag{5b}$$

$$\forall i \in V, \delta'_i = H_i(\delta_i), \tag{5c}$$

where $H_i : \mathbb{R}^3 \to \mathbb{R}^3$ is a transformation specified for the Laplacian coordinates of vertex i, and the superscript $(\cdot)'$ denotes deformed quantities.

Eq. (5a) and Eq. (5b) require that $\mathscr{M}'$ should interpolate the anchors A and handles H. Namely, the deformation should follow the user's direct manipulation. They are, however, not directly contributing to feature preservation.

Eq. (5c) states that $\mathscr{M}$ undergoes deformation $\{H_i\}$, and the mean curvature normal is changed. It is here in Eq. (5c) that further computational requirements are specified to ensure feature preservation. Recall the discussions at the beginning of Section 2 where the key principle for feature preservation is "maximal rigid transformations and minimal shearing." Also recall the approximation property of δ. It is clear that if $\{H_i\}$ is everywhere locally as close as possible to a rigid transformation (*i.e.*, translation, rotation, isotropic scaling), then the deformed mean curvature normal δ'_i will be similar to the initial δ. The resulting $\mathscr{M}'$ will then show

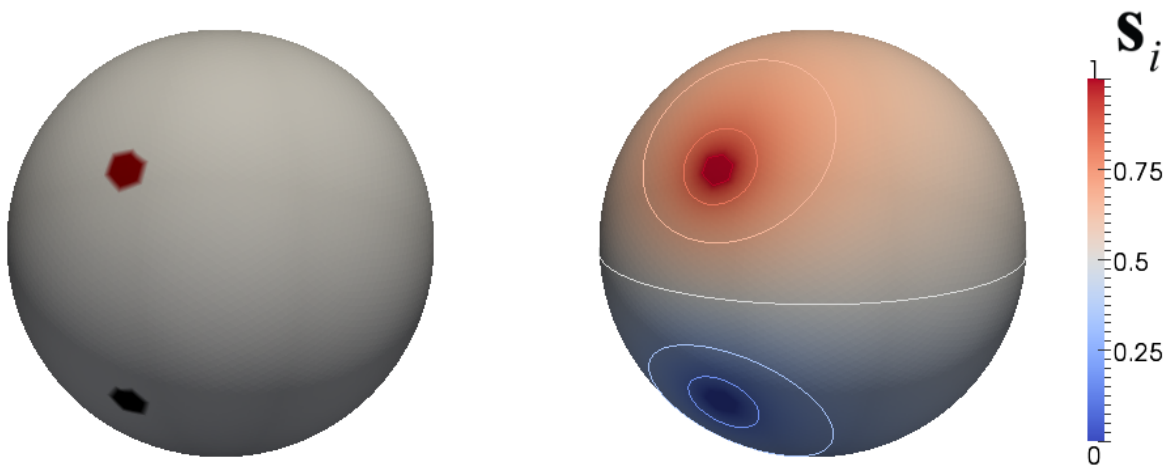


Fig. 4: The smoothly diffused scalar field $\mathbf{s}$ over a sphere. Left: The positions of the anchors A (black), handles H (red). Right: The contour plot of $\mathbf{s}_i$ shows its smooth distribution.

similar visual features as $\mathscr{M}$. This insight (Alexa 2003; Sorkine et al 2004; Lipman et al 2005a) underlies different variants of the Laplacian surface editing algorithms that use different strategies to determine $\{H_i\}$.

The earlier work of (Alexa 2003) assumes $\forall i \in V, H_i = I$ and $\delta' = \delta$. The results are unsatisfactory, because shearing of the features are produced when the δ at the intermediate, unconstrained vertices $\overline{C_A \cup C_H}$ are not rotated to follow C_H. It has been superseded by improved approaches.

Harmonic field (Zayer et al 2005; Masuda et al 2006; Xu et al 2009), a simple and effective approach, is based on the Laplace equations of Eq. (4), quaternion rotation (McDonald 2010) and quaternion interpolation (Shoemake 1985; Johnson 1999). The intuition of harmonic field is to propagate the user-specified deformations at C_H to the un-deformed C_A, such that the deformations $\{H_i, i \in \overline{C_A \cup C_H}\}$ is proportional to their geodesic distances relative to the constrained regions C_A and C_H. It is implemented by creating a smoothly diffused scalar field $\mathbf{s}$ over $\mathscr{M}$ via L, and then using $\mathbf{s}$ as a interpolation parameter to compute H_i. Namely,

$$\begin{aligned}
&L\mathbf{s} = 0 \\
&\text{subject to} \\
&\forall i \in C_A, s_i = 0, \\
&\forall j \in C_H, s_i = 1,
\end{aligned} \tag{6}$$

A typical solution of this equation over a spherical mesh is shown in Figure 4.

To obtain $\{H_i, i \in \overline{C_A \cup C_H}\}$, the deformations at the constrained vertices $\{H_i, i \in C_H\}$ is recovered by a solving Procrustes problem with regard to $\mathbf{v}_i$ and $\mathbf{u}'_i$, or by recording the user's interactions with the manipulator widgets in the user interface (UI). To ensure feature preservation, and in light of the translation invariance of δ, H_i is assumed to be a composition of an isotropic scaling and a rotation. The scaling component is represented by a scalar-valued scale factor, and can be trivially interpolated per s_i. The rotation component is represented by a unit quaternion q_i, and quaternion interpolations are performed per s_i. For the simple case, the user only manipulates one handle region at a time, spherical linear interpolation (Slerp, Shoemake 1985) is used,

$$\begin{aligned} q_i &= \mathrm{Slerp}(q_A, q_H, s_i) \\ &= \frac{\sin\big((1-s_i)\Omega\big)}{\sin(\Omega)} \underbrace{q_A}_{(0,0,0,1)} + \frac{\sin(s_i\Omega)}{\sin(\Omega)} q_H, \end{aligned} \tag{7}$$

where Ω is the great arc from q_A to q_H, and $\cos\Omega = q_A \cdot q_H$. In the case of multiple handles simultaneously, namely shape manipulation with multi-touch input, the multi-variate extension of Slerp (Johnson 1999) is needed.

With $\{H_i\}$ computed, δ'_i can be obtained using Eq. (5c). Then, with the anchors and the handles constraints (Eq. 5a and 5b) determined from the user input, $\mathbf{v}'$ can be reconstructed through the following least-square minimization:

$$\begin{aligned} &\underset{\mathbf{v}'}{\text{minimize}}\ \|L_D\mathbf{v}' - \delta'\|^2 \\ &\text{subject to Eq. (5a) and Eq. (5b).} \end{aligned} \tag{8}$$

In graphics applications, an unconstrained least squares problem is often solved in place of the linearly constrained least squares problem of Eq. (8). The unconstrained problem is obtained by treating the constraints of Eq. (5a) and Eq. (5b) as soft constraints that augment L_D and δ, namely

$$\underset{\mathbf{v}'}{\text{minimize}} \left\| \begin{pmatrix} L_D \\ L_{D,aug} \end{pmatrix} \mathbf{v}' - \begin{pmatrix} \delta' \\ \delta'_{aug} \end{pmatrix} \right\|^2, \tag{9}$$

where, similar to the block diagonal matrix L_D,

$$L_{D,aug} = \begin{pmatrix} L_{aug} & & \\ & L_{aug} & \\ & & L_{aug} \end{pmatrix},$$

with diagonal block

$$L_{aug} = \begin{pmatrix} A_{aug} \\ H_{aug} \end{pmatrix},$$

$$A_{aug} = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ |A| \end{array} \begin{array}{c} \begin{array}{cccccccc} \cdots & A_1 & \cdots & A_2 & \cdots & A_{|A|} & \cdots \end{array} \\ \begin{pmatrix} \cdots & 1 & \cdots & 0 & \cdots & 0 & \cdots \\ \cdots & 0 & \cdots & 1 & \cdots & 0 & \cdots \\ \vdots & 0 & \vdots & 0 & \ddots & 0 & \vdots \\ \cdots & 0 & \cdots & 0 & \cdots & 1 & \cdots \end{pmatrix} \end{array}, \tag{10}$$

and similarly

$$H_{aug} = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ |H| \end{array} \begin{array}{c} \begin{array}{cccccccc} \cdots & H_1 & \cdots & H_2 & \cdots & H_{|H|} & \cdots \end{array} \\ \begin{pmatrix} \cdots & 1 & \cdots & 0 & \cdots & 0 & \cdots \\ \cdots & 0 & \cdots & 1 & \cdots & 0 & \cdots \\ \vdots & 0 & \vdots & 0 & \ddots & 0 & \vdots \\ \cdots & 0 & \cdots & 0 & \cdots & 1 & \cdots \end{pmatrix} \end{array}. \tag{11}$$

Note that $A_{aug} \in \mathbb{R}^{|A|\times n_v}, H_{aug} \in \mathbb{R}^{|H|\times n_v}$, and hence $L_{aug} \in \mathbb{R}^{(|A|+|H|)\times n_v}$. Correspondingly, $\delta'_{aug} \in \mathbb{R}^{3(|A|+|H|)}$, like δ, is a concatenation of the components of $\{\mathbf{v}_i, i \in A\}$ and $\{\mathbf{u}'_i, i \in H\}$ which come from Eq. (5a) and Eq. (5b).

Sorkine et al (2004) described a different approach. Instead of computing the $\{H_i\}$ via harmonic field, they formulated H_i as a linear function of the unknown $\mathbf{v}'$, by first assuming $H_i(\mathbf{v}')$ to have the form of $s\exp(R)$ where $s \in \mathbb{R}$ represents isotropic scaling and $R \in \mathbb{R}^{3\times 3}$ is skew-symmetric, and then linearizing $H_i(\mathbf{v}')$ through the power expansion of the rotation transformation $\exp(R)$. Using this implicit formulation, the objective function optimized to reconstruct $\mathbf{v}'$ becomes

$$\underset{\mathbf{v}'}{\text{minimize}} \; \|L_D\mathbf{v}' - \big(H_i(\mathbf{v}')\big)\delta\|^2$$

subject to Eq. (5a) and (5b),

which has a form similar to Eq. (8) and is solved using soft constraints as well.

The above algorithmic differences between the explicit and implicit formulation have two implications. Firstly, from the perspective of user interactions, the explicit form have a stronger requirement on user input:

the user must translate and re-orient more than two vertices, or specify the desired position as well as normal when manipulating a single vertex. Otherwise, if the user is only translating a single vertex, there is not sufficient information to compute or propagate $H_i, i \in H$. The implicit formulation relaxes such requirements, and the user is able to manipulate a single vertex. Secondly, the explicit form has the lower level algorithmic automation of the two, but lends itself well to increased user controllability. As described later in Section 3.2, the explicit form can be modified to introduce user-customizable, region-dependent deformations.

In summary, the complete Laplacian surface editing algorithm is described in Algorithm 1.

Algorithm 1 Laplacian surface editing (LSE)

```
Compute $L$ per Eq. (3);
Extract $\delta$ per Eq. (1);
Compute $\{H_i\}$, using any approach described in Section 2.1.2;
Compute $\delta'$ per Eq. (5c);
Reconstruct $\mathbf{v}'$ from $\delta'$ per Eq. (1).
```

2.1.3 Other surface deformation algorithms

Other forms of differential coordinates are not based on the discrete Laplacian operator, which have led to alternative mesh editing approaches. Yu et al (2004) have used the gradients of the Cartesian coordinates as the feature-preserving coordinates and derived a mesh editing approach similar to Laplacian surface editing (Sorkine 2006). Zhou et al (2005) have extended the Laplacian approach from surface meshes to volumetric meshes. Sheffer and Kraevoy (2004); Lipman et al (2005b, 2007a) have presented differential coordinates that are rotation invariant and robust under large rotations. Botsch et al (2006); Au et al (2007); Sorkine and Alexa (2007); Eigensatz et al (2008) have studied nonlinear objectives for feature-preservation, in lieu of the linear least squares objectives such as Eq. (8). Their formulations necessitate iterative solving procedures, thus improving the rigid preservation of the features at the cost of computational efficiency. Botsch and Sorkine (2008); Cohen-Or (2009) provides a comprehensive survey.

Unlike surface deformation, where the user directly manipulates individual vertices of $\mathscr{M}$, space deformation utilizes, and thus constrains user manipulations to, an auxiliary mesh $\mathscr{M}_a := (V_a, T_a)$, also known as a lattice or a cage, that fully or partially encloses $\mathscr{M}$ and has a considerably lower vertex count than $\mathscr{M}$, *i.e.*, $|V_a| \ll |V|$. Unlike space deformation, the users are *not* able to directly specify the anchors A, handles H or desired handle positions $\mathbf{u}'$ on $\mathscr{M}$. Instead, they manipulate the ambient space $\mathscr{M}_a$ and specify $\mathbf{v}'_a$ which denotes the deformed position of every vertex in $\mathscr{M}_a$.

The forerunners of feature-preserving space deformation algorithms are a line of space deformation algorithms based on embedded coordinates and *without* feature preservation. An early example is the free-form deformation (FFD, Sederberg and Parry 1986) which have proposed the following workflow shared by subsequent algorithms.

Creating the auxiliary mesh $\mathscr{M}_a$ In FFD, a parallelpiped lattice is created around the geometry under edit $\mathscr{M}$.

Extracting the parametric coordinates η For vertex i of the mesh $\mathscr{M}$, its parametric coordinates η_i are expressed as a vector-valued function of $\mathbf{v}_i$, such that a mapping F from the Cartesian coordinates $\mathbf{v}_{a,j}$ of some lattice vertices to $\mathbf{v}_i$ can be established in the general form of

$$\mathbf{v}_i = F\Big(\eta_i(\mathbf{v}_i), \{\mathbf{v}_{a,j}, j \in \Psi_i\}\Big), \tag{12}$$

where Ψ_i defines the set of $\mathscr{M}_a$ vertices that have influences on the position of mesh vertex i. Specifically in FFD, $\eta_i(\mathbf{v}_i) \in \mathbb{R}^3_{\in[0,1]}$ are the local coordinates within each lattice cell that contains vertex i, Ψ_i includes all the vertices of the lattice cell that contains vertex i, F is the trivariate tensor product Bernstein polynomial.

Deforming the lattice into $\mathbf{v}'_a$ In this step, the user manipulates vertices on the lattice $\mathscr{M}_a$.

Reconstructing $\mathbf{v}'$ Here, the extracted η, Eq. (12) and $\mathbf{v}'_a$ are used in conjunction, such that

$$\mathbf{v}'_i = F\Big(\eta_i(\mathbf{v}_i), \{\mathbf{v}'_{a,j}, j \in \Psi_i\}\Big).$$

Gain and Bechmann (2008) have reviewed the user interaction aspects of many early techniques and remarked on the versatility and ease of use.

For most computer animation applications, it is often desirable to concentrate deformations to the joints of a highly complex character model, in which case FFDs based on a parallelpiped lattices do not suffice. A number of recent, cage-based space deformation algorithms (*e.g.*, Ju et al 2005; Lipman et al 2007b; Joshi et al 2007) are developed, with the following differences from FFD in terms of $\mathscr{M}_a$, η extraction and $\mathbf{v}'$ reconstruction. The auxiliary mesh $\mathscr{M}_a$ in those algorithms are no longer a cuboid lattice, but a free-form, triangulated surface mesh (Ju et al 2005) or tetrahedron volume mesh (Huang et al 2009) surrounding $\mathscr{M}$. The parametric coordinates C_i are expressed as the generalized barycentric coordinates (Meyer et al 2002a) or its variants (Floater 2003; Langer et al 2006), leading the following specialized coordinate extraction and reconstruction equation,

$$\mathbf{v}_i = \sum_{j \in V_a} \eta_i(\mathbf{v}_i)\mathbf{v}_{a,j}, \tag{13}$$

and

$$\mathbf{v}'_i = \sum_{j \in V_a} \eta_i(\mathbf{v}_i)\mathbf{v}'_{a,j}. \tag{14}$$

Lipman et al (2008) have remarked that Eq. (13) and Eq. (14), expressing the mesh coordinates as a linear combination of cage coordinates, are not feature-preserving, because any shearing on the cage geometry will be transferred to the mesh. They propose a different local, parametric coordinates known as the Green coordinates, leading to the corresponding extraction and reconstruction equations as follows,

$$\mathbf{v}_i = \sum_{j \in V_a} \eta_{v,i}(\mathbf{v}_i)\mathbf{v}_{a,j} + \sum_{k \in T_a} \eta_{n,i}(\mathbf{v}_i)\mathbf{n}_{a,k}, \tag{15}$$

and

$$\mathbf{v}'_i = \sum_{j \in V_a} \eta_{v,i}(\mathbf{v}_i)\mathbf{v}'_{a,j} + \sum_{k \in T_a} \eta_{n,i}(\mathbf{v}_i)\mathbf{n}'_{a,k}. \tag{16}$$

A linear combination of the cage face normals $\mathbf{n}_a$ is added in Eq. (15) and Eq. (16). Lipman et al (2008) have derived the closed-form formulas for the coordinates $\eta_{v,i}$ and $\eta_{n,i}$ and proved that the deformed $\mathbf{v}'$ exhibit bounded distortions, *i.e.*, feature preservation. The major steps of this algorithm are summarized in Algorithm 2.

Algorithm 2 Space deformation via Green coordinates (GC)

```
Create $\mathscr{M}_a$ around $\mathscr{M}$;
Extract coordinates $\eta_{v,i}$ and $\eta_{n,i}$ per Lipman et al (2008);
Deform $\mathscr{M}_a$ into $\mathbf{v}'_a$;
Reconstruct $\mathbf{v}'$ from $\mathbf{v}'_a$ per Eq. (16).
```

Compared to feature-preserving surface deformation reviewed in Section 2.1, space deformation has the following advantages. First of all, it can handle a wider range of model representations than surface deformation, including point sets, polygon soups or manifold surface mesh. This is because the extraction and reconstruction of $\mathbf{v}$ do not rely on any topological information T of $\mathscr{M}$. Secondly, the computational efficiency of space deformation is better, especially in the reconstruction step, which involves nothing but vector sum and matrix-vector product and is convenient to exploit computational parallelism. The reconstruction in surface deformation, in contrast, entails solving large, sparse linear systems which is slower (Lipman et al 2008).

Space deformation has a major disadvantage: the user interaction is less intuitive, because the user has to manipulate the ambient cage $\mathscr{M}_a$, not the underlying $\mathscr{M}$. With fast, closed-form formulations such as the Green coordinates, it is impossible to specify positional constraints on the mesh vertices. With subsequent algorithms designed to overcome this difficulty and enable the direct manipulation of vertices of $\mathscr{M}$ (*e.g.*, Botsch et al 2007; Sumner et al 2007; Ben-Chen et al 2009, reviewed in detail later in Section 2.3), iterative optimization such as Newton's method are involved and the computational efficiency is compromised.

The extra pre-processing step unique to space deformation, namely the creation of the auxiliary mesh $\mathscr{M}_a$, though non-trivial, is not a severe disadvantage. Several heuristics for creating $\mathscr{M}_a$ has been proposed, including voxelizing $\mathscr{M}$ (Botsch et al 2007), sampling points from $\mathscr{M}$ (Sumner et al 2007; Ben-Chen et al 2009), or creating a hexagonal pyramid around a region of interest on $\mathscr{M}$ (Li et al 2010). The recent algorithm (Ben-Chen et al 2009) has also reduced the sensitivity to the cage mesh $\mathscr{M}_a$.

2.3 Unifying of surface and space deformations

Algorithm 1 (LSE) and Algorithm 2 (GC) bear overall resemblance in their computational processes. Both extract a per-vertex representation, δ or η, as a function of some influential vertices $\{\mathbf{v}_i, i \in \Psi_i\}$ to encode the geometry features of $\mathscr{M}$. Both reconstruct the deformed Cartesian coordinates $\mathbf{v}'$ from the features. This general paradigm of extract-and-reconstruct has also been used in other general or special purpose editing systems, where it is known as "analysis-and-edit." For instance, Gal et al (2009) introduce a general purpose direct modeling system using wires as handles. Kraevoy et al (2008) describe a grid-based space deformation system specialized for feature preservation during scaling.

Specifically, LSE can be described in terms of a dense space deformation. First, instead of creating the cage $\mathscr{M}_a$ separately, $\mathscr{M}$ itself is used as $\mathscr{M}_a$, therefore the intuitive direct manipulation of $\mathscr{M}$ is enabled. Then, the extraction of the local coordinates corresponds to computing δ using Eq. (1). Finally, the reconstruction from δ involves a least squares minimization, in contrast to the close-form reconstruction that GC has (Eq. 16). This analogy helps to highlight the design trade-offs that underly the pros-and-cons of both algorithms.

Cohen-Or (2009) has advocated the research opportunity of unifying the strengths of both algorithms. Several hybrid deformation systems have been developed in this spirit. Sumner et al (2007); Botsch et al (2007); Ben-Chen et al (2009) create $\mathscr{M}_a$ by sampling a sparse mesh subgraph, voxels or vertices, respectively, from $\mathscr{M}$. The resulting algorithms allow direct manipulation of $\mathscr{M}$, while retaining the efficiency of space deformation. Masuda (2007) use both surface and space deformation jointly to deform assembly models that have multiple components. Each component is deformed by surface deformation, and deformation is propagated between components through space deformation.

3 Proposed approach

3.1 Overview

We formalize the goal of the proposed approach as follows. Compared with the direct modeling problem in Section 2, additional goals arise due to the differences in the intended uses, *i.e.*, computer-aided industrial design *versus* computer graphics and animation. Here we would like to additionally address two objectives.

Aesthetic design intention management Allow the user to exactly preserve designed features in certain regions, and selectively relax feature preservation in a controllable fashion in others. The user should be able to express such intentions via color-coded surface annotations.

Engineering constraints Allow the user to add and enforce dimensional constraints that specify pair-wise distances that are axis-aligned or not.

Our solution to this problem is built upon Laplacian surface editing with harmonic guidance (Zayer et al 2005), because this explicit formulation provides the necessary flexibility to enable region-dependent customization of the deformations Xu et al (2006); Popa et al (2007), and conveniently admits equality constraints Masuda et al (2006). We decompose the problem into the two sub-problems in the following sections, each addressing one objective above.

Relations to existing works There are two major differences from the existing surface deformation algorithms in computer graphics. First, we enable the users to specify different design intentions for different regions, rather than deforming every part of the surface uniformly. Second, we enable the exact enforcement of certain constraints, rather than treating them as soft constraints that are satisfied in a least squares sense (Eq. 9). Those differences are rooted in the different focuses of the intended applications: graphics applications emphasize algorithmic automation and the resulting visual pleasingness, whereas our industrial design application mandates user control, visual pleasingness as well as engineering exactness. The formulations underlying those differences are detailed in Section 3.2 and Section 3.3, respectively.

3.2 Preserving aesthetic design intentions

When the surface deformation algorithm utilizes the harmonic guidance formulation, this objective translates, mathematically, into a modified Eq. (6) with region-dependent gradients for the solution **s**. Intuitively, consider Eq. (6) as a steady-state thermal problem with fixed temperature boundary conditions. In its original form, Eq. (6) describes the temperature distribution over a domain with unit material property. If non-uniform material properties are introduced, it will change the temperature distribution, for instance, concentrating temperature drops in the regions with high thermal resistance. Recall that the temperature field **s** is used to interpolate deformations, then the regions with high thermal resistance will have a large **s** gradient, thus absorbing much deformations propagated from the handles and resulting in large deformations in the region. In the opposite extreme case of a region with low or zero thermal resistance, the per-vertex temperature values **s** within should be equal (*i.e.*, zero gradient), and this nicely corresponds to total rigidity for this region, because the **s**-interpolated per-vertex deformations are the same. This material property metaphor aligns well with the design practice in industrial design which makes frequent use of annotations and surface shadings to express design intents.

To implement the above idea, we additionally define κ which is a per-face property over $\mathscr{M}$. The user is able to assign κ_i to triangle i through a painting widget in the UI. We use κ to modify L into the material-aware

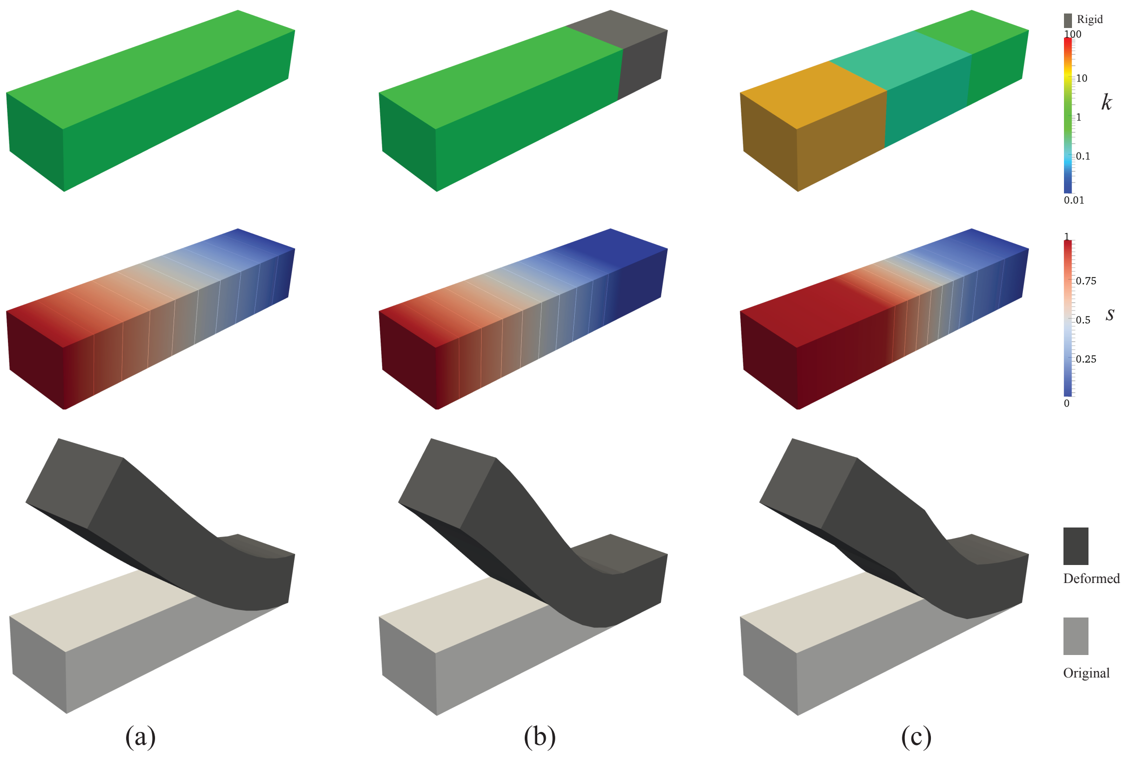


Fig. 5: The bending of a rectangular column with different material properties shown in the first row. In the second row, the contour plot shows the distributions of the deformation interpolation coefficient s. In the third row, the deformed shapes and the original shapes are compared. In all cases, the vertices on the near end are the handles and the vertices on the far end are the anchors. (a) Uniform material property. (b) Non-uniform material property with one rigid region painted in black. (c) Non-uniform material property with varying stiffness.

Laplacian L_κ. With the thermal analogy, at the steady state, the following Laplacian equation should hold

$$\begin{aligned} & L_\kappa \mathbf{s} = 0 \\ & \text{subject to} \\ & \forall i \in C_A, s_i = 0, \\ & \forall j \in C_H, s_i = 1. \end{aligned} \tag{17}$$

Using the thermal equilibrium at the vertex i, L_κ can be determined as

$$(L_\kappa)_{ij} := \begin{cases} -1 & i = j \\ \frac{1}{2}(\kappa_\alpha \cot\alpha_{ij} + \kappa_\beta \cot\beta_{ij}) & j \in N(i) \\ 0 & \text{otherwise,} \end{cases} \tag{18}$$

where κ_α and κ_β are the face properties for the triangles whose sides bound the angles α_{ij} and β_{ij}, respectively. Our derivation differs slightly from Xu et al (2006) whose derivation was based on Poisson mesh editing (Yu et al 2004) and results in κ^2 instead of κ. In large regions with zero thermal resistance, $\kappa = \infty$ leads to numerical stability issues. Instead we specify $\kappa = 5000$ and post-process the solution vector $\mathbf{s}$ to ensure equality within the rigid regions. Figure 5 shows the contours of $\mathbf{s}$ and the deformations after introducing locally rigid regions and varying material properties.

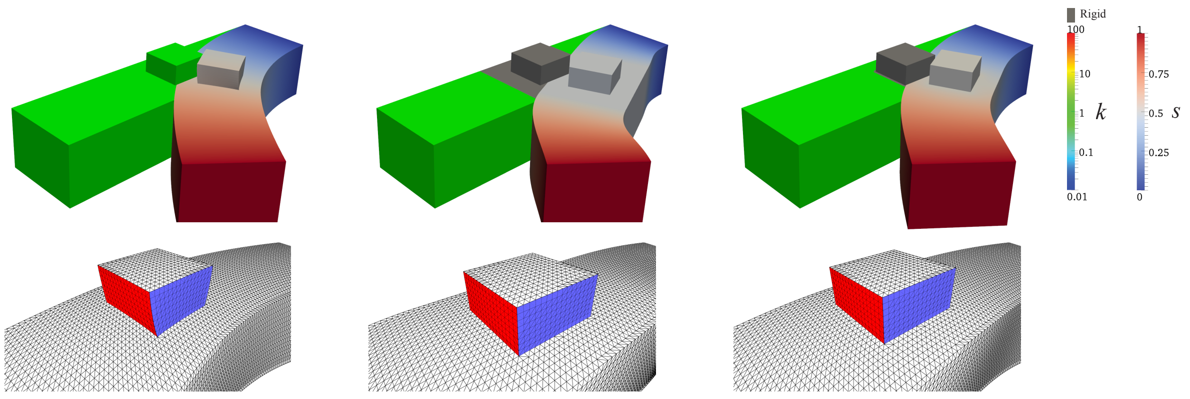


Fig. 6: The deformation results of a column, when different portions are constrained as rigid features. First row: the original shapes and the deformed shapes. The rigid regions, color-coded in black, are indicated by assigning material properties on the original shapes. The distributions of the deformation interpolation coefficient $\mathbf{s}$ are drawn on the deformed shapes. Second row: close-up views of the cuboid features under deformation. Two front faces are colored in red and blue to enhance the visual clarity of the boundaries. Note the distortions, or the lack thereof, in the three examples.

3.3 Enforcing engineering constraints

Recall that in harmonic guidance, the Cartesian coordinates $\mathbf{v}$ are reconstructed through a least squares problem (Eq. 8). Then the objective of enforcing engineering constraints between the vertices can be translated, mathematically, into a constrained least squares problem. Depending on the nature of the dimensional constraints, axis-aligned or not, they can be categorized as linear or quadratic. In the followings sections, we first review the general case of each category, and then proceed to describe detailed constraint formulations that are specific to our problem.

3.3.1 Linear least squares with linear constraints

The problem of linearly constrained least squares, in its general form (Golub and Van Loan 1996; Björck 1996; Laratta and Zironi 2001), is

$$\begin{aligned}&\underset{\mathbf{x}}{\text{minimize}}\ \|A\mathbf{x}-\mathbf{b}\|^2\\&\text{subject to } C\mathbf{x}=\mathbf{d},\end{aligned}\tag{19}$$

where $\mathbf{x}\in\mathbb{R}^n, A\in\mathbb{R}^{m\times n}, \mathbf{b}\in\mathbb{R}^m, C\in\mathbb{R}^{p\times n}, \mathbf{d}\in\mathbb{R}^p$.

The Lagrange function of Eq. (19) is

$$\mathscr{L}(\mathbf{x},\lambda)=\|A\mathbf{x}-\mathbf{b}\|^2+2\lambda^{\mathrm{T}}(C\mathbf{x}-\mathbf{d}),\tag{20}$$

where $\lambda\in\mathbb{R}^p$ are the Lagrange multipliers. Rearranging $\frac{\partial\mathscr{L}}{\partial\mathbf{x}}=0$ gives an equivalent, unconstrained equation

$$\begin{pmatrix}A^{\mathrm{T}}A & C^{\mathrm{T}}\\ C & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \lambda\end{pmatrix}=\begin{pmatrix}A^{\mathrm{T}}\mathbf{b}\\ \mathbf{d}\end{pmatrix}\tag{21}$$

which can be solved directly or iteratively for $\mathbf{x}$.

In our approach, $A=L_D, \mathbf{x}=\mathbf{v}', \mathbf{b}=\delta, m=n=3n_v$. Three types of linear equality constraints are involved and constitute the building blocks of C and $\mathbf{d}$. First of all, the anchor and handle constraints from Eq. (8) are exactly enforced. A_{aug} (Eq. 10) and H_{aug} (Eq. 11) and the corresponding $\boldsymbol{\delta}_{\mathbf{aug}}$ are appended to C and $\mathbf{d}$.

Secondly, as in Masuda et al (2006), linear constraints are applied to the vertex triplet $\{(i, j, k) \in T_{rigid}\}$ inside each rigid triangle T_{rigid} (*i.e.*, one with zero “thermal resistance”) to ensure rigidity, namely

$$\begin{cases} \mathbf{v}'_i - \mathbf{v}'_j = H_i(\mathbf{v}_i - \mathbf{v}_j) \\ \mathbf{v}'_j - \mathbf{v}'_k = H_i(\mathbf{v}_j - \mathbf{v}_k) \end{cases}, \tag{22}$$

where H_i is the interpolated per-vertex transformations via harmonic guidance. In the rigid region, $H_i = H_j = H_k$, as is dictated by Eq. (17). Eq. (22) and Eq. (17) together guarantees that the rigid region are only subject to rigid or similar transformations, not shearing.

Masuda et al (2006) have noted that specifying the pair-wise constraints (Eq. 22) for $\{(i, j, k)\}$ in every triangle in the rigid region is redundant. Instead, a minimal spanning tree is extracted from the rigid region and a reduced set of constraints is applied to any two adjacent vertices in the spanning three.

Finally, the axis-aligned distance vector between vertex i and j, $\mathbf{d}_{ij} = (d^x_{ij}, d^y_{ij}, d^z_{ij})$, can be written as a linear constraint. It takes the form of

$$\mathbf{v}'_i - \mathbf{v}'_j = \mathbf{d}_{ij}. \tag{23}$$

The the left-hand side of Eq. (22) and Eq. (23) share the same structure. Both can be rearranged into the matrix notation [1]

$$\begin{pmatrix} C_{i-j} & & \\ & C_{i-j} & \\ & & C_{i-j} \end{pmatrix} \mathbf{v}'$$

where the diagonal block is a row vector

$$C_{i-j} = \begin{array}{c} \\ 1 \end{array} \begin{pmatrix} \cdots & i & \cdots & j & \cdots \\ 0 & 1 & 0 & -1 & 0 \end{pmatrix} \in \mathbb{R}^{1 \times n_v}$$

which, together with A_{aug} (Eq. 10) and H_{aug} (Eq. 11), populates the left-hand side of the equality constraint equation $C\mathbf{x} = \mathbf{d}$. And correspondingly, $H_i(\mathbf{v}_i - \mathbf{v}_j)$ or the column vector $\mathbf{d}^{\mathrm{T}}_{ij}$ are appended to the right-hand side of the constraint equation to assemble the full constraint equality. Multiple pairs of such constraints can be added, as long as they are consistent.

[1] We use C_{i-j} to denote the constraint coefficients when specifying the pair-wise relationship between vertex i and j. We wish to note that C_{i-j} is different from C_{ij} which mean the (i, j)-indexed element of C.

Figure 6 shows examples of rigidity-constrained deformation solved using the linearly constrained least squares formulation, in which user-specified feature regions are preserved.

3.3.2 Linear least squares with quadratic constraints

The quadratically constrained linear least squares problem (Gander 1981; Golub and von Matt 1991; Chan et al 1992; Schöne and Hanning 2011) takes the general form of

$$\begin{aligned}&\underset{\mathbf{x}}{\text{minimize}}\ \|A\mathbf{x}-\mathbf{b}\|^2\\&\text{subject to } \mathbf{x}^{\mathrm{T}}C\mathbf{x}=d,\end{aligned}\tag{24}$$

where $\mathbf{x}\in\mathbb{R}^n, A\in\mathbb{R}^{m\times n}, \mathbf{b}\in\mathbb{R}^m, C\in\mathbb{R}^{n\times n}, d\in\mathbb{R}$.

The Lagrange function of Eq. (24) is

$$\mathscr{L}(\mathbf{x},\lambda)=\|A\mathbf{x}-\mathbf{b}\|^2+\lambda(\mathbf{x}^{\mathrm{T}}C\mathbf{x}-d),\tag{25}$$

where $\lambda\in\mathbb{R}$ is the Lagrange multiplier. The partial derivatives $\frac{\partial\mathscr{L}}{\partial(\cdot)}$ are

$$\frac{\partial\mathscr{L}}{\partial\mathbf{x}}=2(A^{\mathrm{T}}A\mathbf{x}-A^{\mathrm{T}}\mathbf{b})-2\lambda C\mathbf{x},\tag{26a}$$

$$\frac{\partial\mathscr{L}}{\partial\lambda}=\mathbf{x}^{\mathrm{T}}C\mathbf{x}-d\tag{26b}$$

which lead to the normal equations

$$(A^{\mathrm{T}}A+\lambda C)\mathbf{x}=A^{\mathrm{T}}\mathbf{b},\tag{27a}$$

$$\mathbf{x}^{\mathrm{T}}C\mathbf{x}=d.\tag{27b}$$

Chan et al (1992) point out that if the unknowns $\mathbf{x}$ are treated as an intermediate variable $\mathbf{x}(\lambda)$ that is dependent on λ, then Eq. (27) becomes a rational equation with the sole unknown λ, and can be solved iteratively using Newton's method. The λ-iteration starts with $\lambda=0$ or some estimates Golub and von Matt (1991); Chan et al (1992). It alternates between two phases: 1) substituting λ into Eq. (27a), solve for $\mathbf{x}(\lambda)$ in a least squares sense; 2) substituting $\mathbf{x}(\lambda)$ into Eq. (27b), solve for λ that reduces $f(\lambda):=\big(\mathbf{x}(\lambda)^{\mathrm{T}}C\mathbf{x}(\lambda)-d\big)$ towards 0. The λ-iteration terminates when $f(\lambda)$ is within tolerance. Once λ is solved, it is substituted back into Eq. (27a) to solve for $\mathbf{x}$.

In our approach, $A = L_D, \mathbf{x} = \mathbf{v}', \mathbf{b} = \delta, m = n = 3n_v$. The quadratic constraint is used when specifying the distance scalar d_{ij} between vertex i and j that are not necessarily axis-aligned, and with unspecified x, y, z components. Such Euclidian distance constraints can be expressed as

$$\|\mathbf{v}'_i - \mathbf{v}'_j\| = d_{ij},$$

or in matrix notation

$$\mathbf{v}'^{\mathrm{T}} \begin{pmatrix} C_{i-j} & & \\ & C_{i-j} & \\ & & C_{i-j} \end{pmatrix} \mathbf{v}' = d_{ij}^2,$$

where the repeating diagonal block is the sparse matrix

$$C_{i-j} = \begin{array}{c} \\ \vdots \\ i \\ \vdots \\ j \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccc} \cdots & i & \cdots & j & \cdots \end{array} \\ \left(\begin{array}{ccccc} & & & & \\ & 1 & & -1 & \\ & & \ddots & & \\ & -1 & & 1 & \\ & & & & \end{array} \right) \end{array}.$$

Figure 7 shows an example of quadratically constrained editing of a unit sphere. The south pole and the vertices on the equator of the sphere are anchored, while the distance between the north and south poles are increased to a specified value. The resulting shape satisfies the constraints up to a relative error of 10^{-4}.

3.4 User interface

The user interface of our editing system is prototyped as a plug-in for OpenFlipper (Moebius and Kobbelt 2010), which provides a wide range of necessary functionalities such as file operations, mesh rendering, vertex selection, vertex manipulation, and face painting. Figure 8 shows the main UI and the trackball widget used for manipulating handle vertices. This UI is meant to be a prototype, and therefore is not thoroughly optimized for user experience.

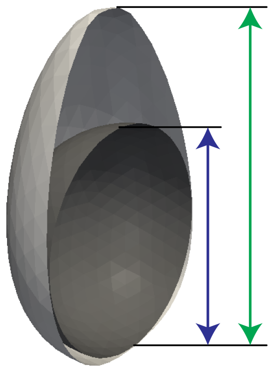
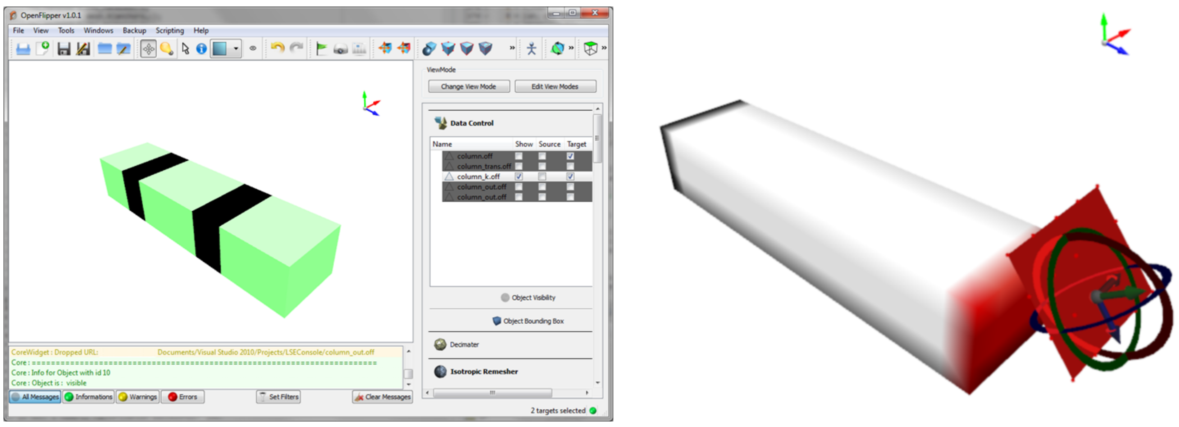


Fig. 7: The deformation of a sphere using dimensional constraints. The south pole and the vertices on the equator of the sphere are anchored, while the distance between the north and south poles are increased from the blue dimension to the green dimension. Dark mesh shows the original sphere. Light mesh shows the deformed shape. The view is clipped by the *yz* plane for visual clarity.



Name	Show	Source	Target
column.off	☐	☐	☑
column_trans.off	☐	☐	☐
column_k.off	☑	☐	☑
column_out.off	☐	☐	☐
column_out.off	☐	☐	☐

```
CoreWidget : Dropped URL:                Documents/Visual Studio 2010/Projects/LSEConsole/column_out.off
Core : ========================================================================
Core : Info for Object with id 10
Core : Object is :  visible
```

Fig. 8: The main graphical user interface and widgets for direct manipulation. Left: Black paints are applied to the middle sections of a column to indicate rigid regions while deforming. Other regions, painted in light green, have the default deformation behavior. Right: The black region at the far end of the column are the fixed anchors. The red region at the near end are the movable handle vertices. The red surface with a 6-degree-of-freedom triad widget attached indicate the new position of the handle vertices.

4 Examples

Here we showcase three application scenarios using our approach. The first example shows in Figure 9 a lounge chair inspired by the Vitra Amoebe Highback (Panton 1970). A regular profile is massaged, by direct

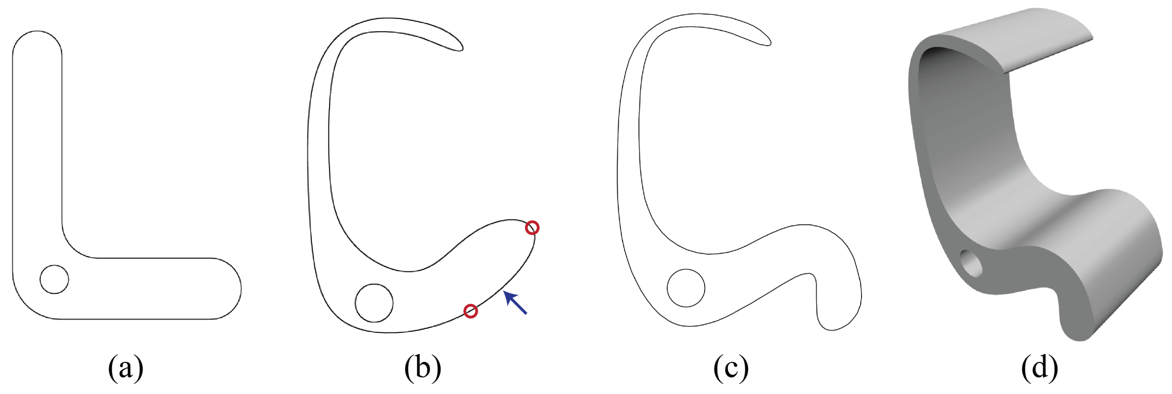


Fig. 9: Direct modeling of a free-form lounge chair. (a) The 2D side-view profile of the starting concept contains very few free-form features. (b) Through a combination of direct manipulation and constraints, the side-view profile is re-styled. (c) The final profile. (d) An extrusion of the profile produces a 3D model of the design.

modeling, into the freeform profile and then extruded to obtain the final 3D model. The second example, shown in Figure 10, illustrates the process of re-styling a mouse to obtain an asymmetric concept out of a generic template. The final example showcases the shape exploration process via direct modeling, in which several variations of the original hand grip are synthesized while preserving the local feature of a text engraving. In all case, features are created via direct manipulation of vertices, and are preserved or relaxed via different levels of rigidity. Engineering constraints, such as the axis-aligned bounding box size or distance between reference points, are also included.

For all the examples, the scale of the model are at the magnitude of 1 unit, typical mesh edge length are at the level of 0.1 unit, and the worst absolute error for constraints violations, which happen at the quadratically constrained vertices, are at the level of 10^{-4}. At the anchors and the handles, the error of constraint violations are usually at the level of 10^{-9}. The total runtime, including the time for constructing the L matrix, solving for the harmonic field $\mathbf{s}$, propagating $\{H_i\}$ and reconstructing $\mathbf{v}'$ are within 1.0 second for a dense mesh with $n_v \leq 10^4$, if only linear constraints are specified. When iteratively solving quadratic constraints, the time taken may go up to 10.0 seconds. With the relatively small meshes used in the examples, the deformations are computed at interactive speed.

We have also identified the following limitations of our approach. Firstly, the iterative process of solving quadratically constrained least squares (Section 3.3.2) is essentially solving a linear version (Section 3.3.1)

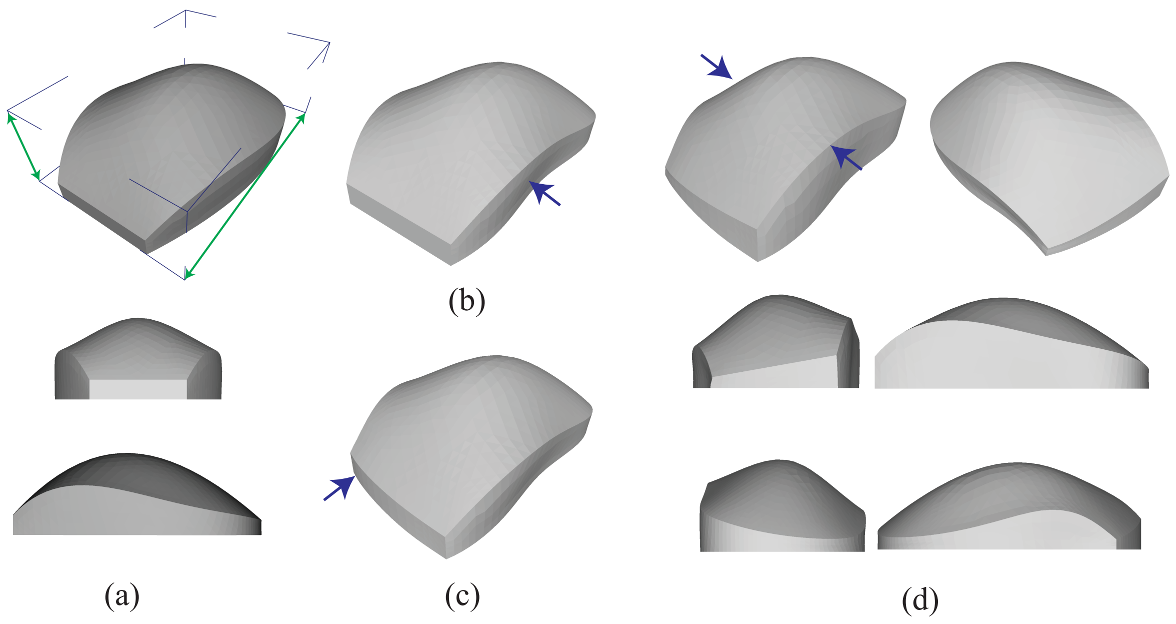


Fig. 10: Direct modeling of a free-form mouse. (a) The initial model features a symmetric concept. (b) The contour of the right side wall is re-styled to create a larger thumb rest. (c) The contour of the front wall is re-designed. (d) Finally, the slant of the top surface is adjusted such that the left portion is higher than the right portion, resulting in an asymmetric concept. The green arrows in (a) indicate the bounding box dimensions that are constrained. The blue arrows in (b), (c) and (d) indicate the design changes at each step.

for several times. Experiments with different meshes suggest that the computational efficiency deteriorates as the mesh size grow, to the extent that the λ-iteration terminates prematurely due to excessive steps. Further efforts are needed to incorporate fast solvers suitable for large sparse matrices such as L_D and C_{i-j}. Secondly, some combinations of anchor and handle selections and rigidity constraints can produce results that are visual unsatisfactory, albeit algorithmically correct. Figure 12 shows an example in this regard. The user has assigned a rigid region that bridges a large portion of the finger grooves between the anchors (top face) and the handles (bottom face). Per the formulation of the material-aware harmonic guidance, most of the deformations will occur at the narrow band of non-rigid, unconstrained vertices near the top and bottom of the hand grip, causing the abrupt, local fluctuations of the contour of the front side. Finally, the ultimate measure of success of the proposed approach is how it boosts the productivity of industrial designers when deployed in lab or field studies, which is more meaningful than the runtime and diagnostic statistics. User studies will be pursued in the future.

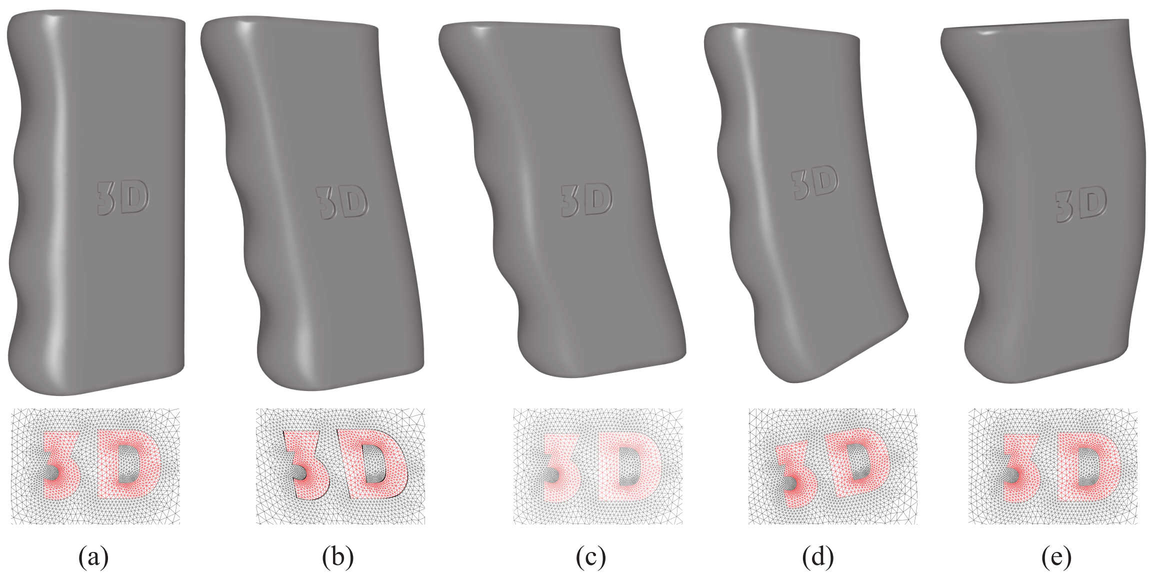


Fig. 11: Direct modeling of the hand grip portion of a hand-held device. (a) The initial model shows a straightly extruded concept, with existing detailed features such as the text engravings and finger grooves. (b) The engraved text and finger grooves are distorted, if the model is deformed without feature preservation. (c), (d) and (e): Using our approach, different alternatives are generated while preserving existing features. The zoomed-in, local mesh wireframes around the text engravings are shown below the shaded models. In (b), the vertical walls of the text engravings are distorted and no longer parallel to the viewing direction. In all other cases, the text engravings are preserved.

5 Concluding remarks

In this paper, we first provide an extensive review of the state-of-the-art direct modeling approaches for 3D mesh-based, freeform surfaces, with an emphasis on the two broad categories of shape deformation algorithms developed in the relevant field of computer graphics: surface deformation and space deformation. We list the relative strengths and weakness of both, and also refer to recent works that tackle the research opportunity of combining the frameworks and the strengths of both.

We then describe a novel 3D surface editing system capable of jointly accommodating aesthetic design intentions expressed in the form of surface painting and color-coded annotations, as well as engineering constraints expressed as dimensions. It is intended for conceptual design that involves frequent shape tuning,

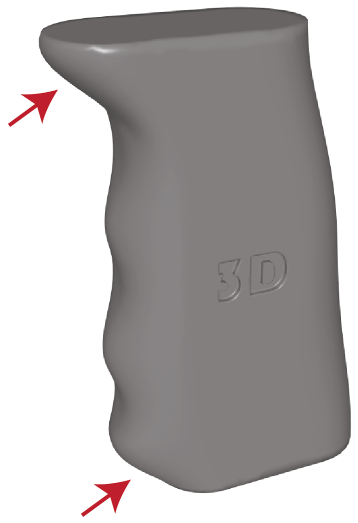


Fig. 12: An unsatisfactory deformation produced if the designer assigns too much rigidity to the finger grooves. Note the abrupt, local fluctuation of the contour of the front side, as indicated by the red arrows.

explorations and the negotiations of engineering and aesthetic considerations, which is still a key challenge for existing direct modeling CAD packages.

The proposed system is built upon Laplacian coordinates, a surface editing approach intended for computer graphics applications. Two issues have prevented us from readily embracing the original form of Laplacian surface editing (LSE) in our target scenarios: firstly, LSE is overly automatic and treats every region of the geometry uniformly, whereas designers wish for a fine-grained control over different regions; secondly, LSE is approximate and satisfies various constraints in a least squares sense, whereas designers wish to prescribe positional constraints and preserve existing aesthetic features exactly. Both issues have been deemed acceptable and received little or no attention in graphics research. Here we address those barriers by incorporating several design-oriented features, such as a region-dependent material property that affects the local deformation behavior (Section 3.2), as well as linearly and quadratically constrained reconstructions from the Laplacian coordinates (Section 3.3). The use of our approach in a few direct modeling scenarios is demonstrated with a prototypical implementation. We note, in particular, that the quadratically constrained case introduces a steep algorithmic challenge in the case of very large meshes, and prompts us to investigate more efficient formulations, such as modified Laplacians based on a sparsely sampled subset of vertices, or subdivision-based representations.

We hope that our approach builds the algorithmic foundation of what we hypothesize provides a higher level of creative support than the current direct modeling CAD systems. We will continue to pursue algorithmic and UI improvements, and conduct user studies to test such hypothesis.

Acknowledgements This section is intentionally blank for review.

References

Alexa M (2003) Differential coordinates for local mesh morphing and deformation. The Visual Computer 19(2):105–114

Alexa M, Wardetzky M (2011) Discrete laplacians on general polygonal meshes. ACM Trans Graph 30(4):102:1–102:10, DOI 10.1145/2010324.1964997

Au OKC, Fu H, Tai CL, Cohen-Or D (2007) Handle-aware isolines for scalable shape editing. ACM Trans Graph 26(3):83:1–83:10, DOI 10.1145/1276377.1276481

Autodesk (2012) Inventor Fusion. URL `http://labs.autodesk.com/technologies/fusion/`

Ben-Chen M, Weber O, Gotsman C (2009) Variational harmonic maps for space deformation. ACM Trans Graph 28(3):34:1–34:11, DOI 10.1145/1531326.1531340

Björck Å (1996) Numerical Methods for Least Squares Problems. SIAM, Philadelphia

Botsch M, Sorkine O (2008) On linear variational surface deformation methods. IEEE Transactions on Visualization and Computer Graphics 14(1):213–230, DOI 10.1109/TVCG.2007.1054

Botsch M, Pauly M, Gross M, Kobbelt L (2006) Primo: coupled prisms for intuitive surface modeling. In: Proceedings of the fourth Eurographics symposium on Geometry processing, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SGP ’06, pp 11–20

Botsch M, Pauly M, Wicke M, Gross M (2007) Adaptive space deformations based on rigid cells. Computer Graphics Forum 26(3):339–347, DOI 10.1111/j.1467-8659.2007.01056.x

Chan TF, Olkin JA, Cooley DW (1992) Solving quadratically constrained least squares using black box solvers. BIT 32(3):481–495, DOI 10.1007/BF02074882

Chen Y, Davis TA, Hager WW, Rajamanickam S (2008) Algorithm 887: CHOLMOD, supernodal sparse cholesky factorization and update/downdate. ACM Trans Math Softw 35(3):22:1–22:14, DOI 10.1145/1391989.1391995

Cohen-Or D (2009) Space deformations, surface deformations and the opportunities in-between. J Comput Sci Technol 24(1):2–5, DOI 10.1007/s11390-009-9200-0

Davis TA (2004) Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. ACM Trans Math Softw 30(2):196–199, DOI 10.1145/992200.992206

Duff IS (2004) MA57—a code for the solution of sparse symmetric definite and indefinite systems. ACM Trans Math Softw 30(2):118–144, DOI 10.1145/992200.992202

Eigensatz M, Sumner RW, Pauly M (2008) Curvature-domain shape processing. Computer Graphics Forum 27(2):241–250, DOI 10.1111/j.1467-8659.2008.01121.x

Floater MS (2003) Mean value coordinates. Comput Aided Geom Des 20(1):19–27, DOI 10.1016/S0167-8396(02)00002-5

Gain J, Bechmann D (2008) A survey of spatial deformation from a user-centered perspective. ACM Trans Graph 27(4):107:1–107:21, DOI 10.1145/1409625.1409629

Gal R, Sorkine O, Mitra NJ, Cohen-Or D (2009) iWIRES: an analyze-and-edit approach to shape manipulation. ACM Transactions on Graphics 28(3):33:1–33:10, DOI 10.1145/1531326.1531339

Gander W (1981) Least squares with a quadratic constraint. Numerische Mathematik 36:291–307

Golub GH, von Matt U (1991) Quadratically constrained least squares and quadratic problems. Numerische Mathematik 59(1):561–580, 10.1007/BF01385796

Golub GH, Van Loan CF (1996) Matrix computations (3rd ed.). Johns Hopkins University Press, Baltimore, MD, USA

Grinspun E, Gingold Y, Reisman J, Zorin D (2006) Computing discrete shape operators on general meshes. Computer Graphics Forum 25(3):547–556, DOI 10.1111/j.1467-8659.2006.00974.x

Huang J, Chen L, Liu X, Bao H (2009) Efficient mesh deformation using tetrahedron control mesh. Comput Aided Geom Des 26(6):617–626, DOI 10.1016/j.cagd.2008.12.002

Johnson MP (1999) Multi-dimensional quaternion interpolation. In: ACM SIGGRAPH 99 Conference abstracts and applications, ACM, New York, NY, USA, SIGGRAPH ’99, p 258, DOI 10.1145/311625.312158

Joshi P, Meyer M, DeRose T, Green B, Sanocki T (2007) Harmonic coordinates for character articulation. ACM Trans Graph 26(3):71:1–71:9, DOI 10.1145/1276377.1276466

Ju T, Schaefer S, Warren J (2005) Mean value coordinates for closed triangular meshes. ACM Trans Graph 24(3):561–566, DOI 10.1145/1073204.1073229

Kraevoy V, Sheffer A, Shamir A, Cohen-Or D (2008) Non-homogeneous resizing of complex models. ACM Transactions on Graphics 27(5):111:1–111:9, DOI 10.1145/1409060.1409064

Langer T, Belyaev A, Seidel HP (2006) Spherical barycentric coordinates. In: Proceedings of the fourth Eurographics symposium on Geometry processing, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SGP ’06, pp 81–88

Laratta A, Zironi F (2001) Computation of lagrange multipliers for linear least squares problems with equality constraints. Computing 67(4):335–350, DOI 10.1007/s006070170004

Li Z, Levin D, Deng Z, Liu D, Luo X (2010) Cage-free local deformations using green coordinates. Vis Comput 26(6-8):1027–1036, DOI 10.1007/s00371-010-0438-x

Lipman Y, Sorkine O, Alexa M, Cohen-or D, Levin D, Rssl C, peter Seidel H (2005a) Laplacian framework for interactive mesh editing. International Journal of Shape Modeling 11(1):43–62, DOI 10.1142/S0218654305000724

Lipman Y, Sorkine O, Levin D, Cohen-Or D (2005b) Linear rotation-invariant coordinates for meshes. ACM Trans Graph 24(3):479–487, DOI 10.1145/1073204.1073217

Lipman Y, Cohen-Or D, Gal R, Levin D (2007a) Volume and shape preservation via moving frame manipulation. ACM Trans Graph 26(1):5:1–5:14, DOI 10.1145/1189762.1189767

Lipman Y, Kopf J, Cohen-Or D, Levin D (2007b) Gpu-assisted positive mean value coordinates for mesh deformations. In: Proceedings of the fifth Eurographics symposium on Geometry processing, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp 117–123

Lipman Y, Levin D, Cohen-Or D (2008) Green coordinates. ACM Trans Graph 27(3):78:1–78:10, DOI 10.1145/1360612.1360677

Masuda H (2007) Feature-preserving deformation for assembly models. Computer-Aided Design and Applications 4(1–4):311–320

Masuda H, Yoshioka Y, Furukawa Y (2006) Interactive mesh deformation using equality-constrained least squares. Computers and Graphics 30(6):936–946, DOI 10.1016/j.cag.2006.08.012

McDonald J (2010) Teaching quaternions is not complex. Computer Graphics Forum 29(8):2447–2455, DOI 10.1111/j.1467-8659.2010.01756.x

Meyer M, Barr A, Lee H, Desbrun M (2002a) Generalized barycentric coordinates on irregular polygons. J Graph Tools 7(1):13–22

Meyer M, Desbrun M, Schröder P, Barr AH (2002b) Discrete differential-geometry operators for triangulated 2-manifolds. In: Visualization and Mathematics III, Springer-Verlag, Berlin, Germany, pp 35–57

Moebius J, Kobbelt L (2010) OpenFlipper: An open source geometry processing and rendering framework. In: Proceedings of Eighth International Conference on Mathematical Methods for Curves and Surfaces, MMCS 2010, Avignon, France

Panton V (1970) Amoebe highback. URL `http://www.vitra.com/en-un/home/products/amoebe-highback/overview/`

Pinkall U, Polthier K (1993) Computing discrete minimal surfaces and their conjugates. Experiment Math 2(1):15–36

Popa T, Julius D, Sheffer A (2007) Interactive and linear material aware deformations. International Journal of Shape Modeling 13(1):73–100

Schöne R, Hanning T (2011) Least squares problems with absolute quadratic constraints. Journal of Applied Mathematics In Press

Sederberg TW, Parry SR (1986) Free-form deformation of solid geometric models. In: Proceedings of the 13th annual conference on Computer graphics and interactive techniques, ACM, New York, NY, USA, SIGGRAPH '86, pp 151–160, DOI 10.1145/15922.15903

Sheffer A, Kraevoy V (2004) Pyramid coordinates for morphing and deformation. In: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium, IEEE Computer Society, Washington, DC, USA, 3DPVT '04, pp 68–75, DOI 10.1109/3DPVT.2004.99

Shoemake K (1985) Animating rotation with quaternion curves. In: Proceedings of the 12th annual conference on Computer graphics and interactive techniques, ACM, New York, NY, USA, SIGGRAPH '85, pp 245–254, DOI 10.1145/325334.325242

Sorkine O (2006) Differential representations for mesh processing. Computer Graphics Forum 25(4):789–807, DOI 10.1111/j.1467-8659.2006.00999.x

Sorkine O, Alexa M (2007) As-rigid-as-possible surface modeling. In: Proceedings of the fifth Eurographics symposium on Geometry processing, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp 109–116

Sorkine O, Cohen-Or D, Lipman Y, Alexa M, Rössl C, Seidel HP (2004) Laplacian surface editing. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, ACM, New York, NY, USA, SGP '04, pp 175–184, DOI 10.1145/1057432.1057456

Sumner RW, Schmid J, Pauly M (2007) Embedded deformation for shape manipulation. ACM Trans Graph 26(3):80:1–80:8, DOI 10.1145/1276377.1276478

Wardetzky M, Mathur S, Kälberer F, Grinspun E (2007) Discrete laplace operators: no free lunch. In: Proceedings of the fifth Eurographics symposium on Geometry processing, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp 33–37

Xu D, Zhang H, Bao H (2006) Non-uniform differential mesh deformation. In: Computer Graphics International, pp 54–65, DOI 10.1007/11784203_5

Xu G (2004) Discrete laplace-beltrami operators and their convergence. Comput Aided Geom Des 21(8):767–784, DOI 10.1016/j.cagd.2004.07.007

Xu K, Zhang H, Cohen-Or D, Xiong Y (2009) Dynamic harmonic fields for surface processing. Comput Graph 33(3):391–398, DOI 10.1016/j.cag.2009.03.022

Yu Y, Zhou K, Xu D, Shi X, Bao H, Guo B, Shum HY (2004) Mesh editing with poisson-based gradient field manipulation. ACM Trans Graph 23(3):644–651, DOI 10.1145/1015706.1015774

Zayer R, Rössl C, Karni Z, Seidel HP (2005) Harmonic guidance for surface deformation. Computer Graphics Forum 24(3):601–609, DOI 10.1111/j.1467-8659.2005.00885.x

Zhou K, Huang J, Snyder J, Liu X, Bao H, Guo B, Shum HY (2005) Large mesh deformation using the volumetric graph laplacian. ACM Trans Graph 24(3):496–503, DOI 10.1145/1073204.1073219