EUROGRAPHICS 2014 / B. Lévy and J. Kautz
(Guest Editors)

Volume 33 (2014), Number 2

Deformation with Enforced Metrics on Length, Area and Volume

Shuo Jin, Yunbo Zhang and Charlie C.L. Wang†

Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong, China

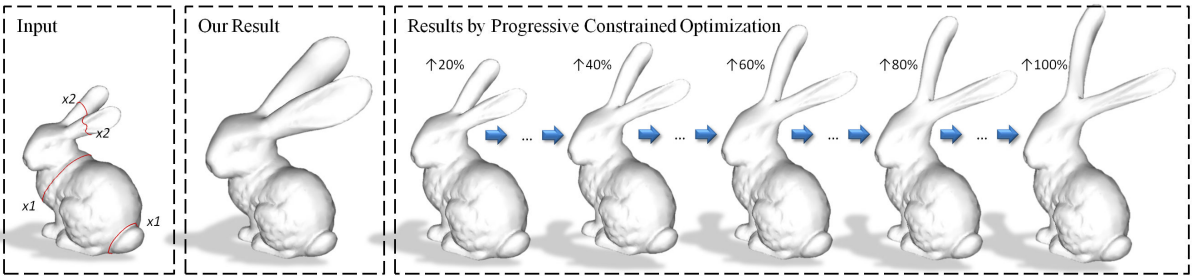

Figure 1: *Our framework for generating deformation with enforced metrics provides a user-friendly tool for designers to accurately control the metrics in a deformation while well preserving the shape of input models. This function is hard to be realized by constrained deformation. Progressive deformation (by gradually increasing the constrained length by 1% in each step) cannot generate results as good as ours. Moreover, the formulation of scale-driven deformation investigated in this work can converge in a few iterations.*

Abstract

Techniques have been developed to deform a mesh with multiple types of constraints. One limitation of prior methods is that the accuracy of demanded metrics on the resultant model cannot be guaranteed. Adding metrics directly as hard constraints to an optimization functional often leads to unexpected distortion when target metrics differ significant from what are on the input model. In this paper, we present an effective framework to deform mesh models by enforcing demanded metrics on length, area and volume. To approach target metrics stably and minimize distortion, an iterative scale-driven deformation is investigated, and a global optimization functional is exploited to balance the scaling effect at different parts of a model. Examples demonstrate that our approach provides a user-friendly tool for designers who are used to semantic input.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—

1. Introduction

Mesh deformation is an important technique in geometric modeling and processing and has many applications in computer graphics. Effective and efficient deformation approaches are also demanded in engineering design of a variety of industries. Given an input 3D model, users often need to modify and deform the model according to different scenarios of usage, i.e. different constraints will be added to the model based on different application demands. Many mesh deformation techniques use handles as input for designers (e.g., [BK04,LSCO*04,LSLCO05,VFTS06,BS08]). Although it is flexible, the handles are hard to be controlled to let the deformed model satisfying certain metrics (e.g., length, area and volume). The authors of [EP09] attempted to tackle this problem by converting the metrics into dif-

† Corresponding Author; E-mail: cwang@mae.cuhk.edu.hk

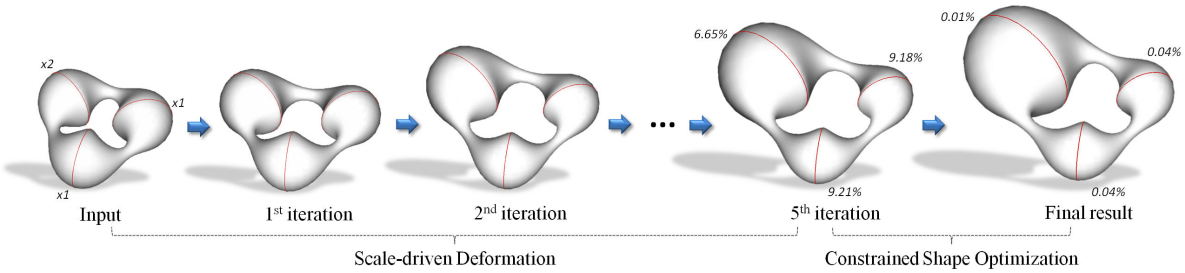

Figure 2: *An illustration of work flow. The input mesh is prescribed with three length constraints. The model is gradually deformed towards the user-specified values on the length constraints with the help of the scale-driven deformation (Section 2). When the lengths are close to the demanded values, constrained shape optimization is applied to enhance the metrics finally (Section 2.4). Different types of metrics, including length, area and volume (Section 3), can be enforced by our framework.*

ferent linear/non-linear constraints to be integrated into the functional of shape optimization. However, how to accurately enforce the constraints of various metrics during deformation is still an open problem.

Constrained deformation is often formulated under the framework of non-linear optimization. Prior approaches convert constraints into a few terms of functional to be minimized. As discussed in [BS08], one has to choose a sufficiently large weight for the soft constraints to obtain satisfactory results, which unfortunately may lead to numerical problems. Using hard constraints to fulfill the demanded metrics leads to the results with unexpected distortion. The reason is as follows. In this case, the input model to be deformed does not present the demanded values on the metrics. Therefore, the initial guess of optimization actually falls outside the feasible region defined by the hard constraints. Applying Lagrange multiplier directly on the constraints projects the initial guess onto a near point inside the feasible region, which does not give an explicit consideration of shape preservation (see Fig.1). This is different from those volume-preserving deformation techniques (e.g., [HML00, VFTS06, AB97, RSB96]), where the input model has already been inside the feasible region. To keep a deformed mesh being close to the feasible region, one may consider the strategy of progressively strengthening the demand on metrics. Ideally, when the instant shape is sufficiently close to the feasible region, enforcing the demanded metrics as hard constraints will give small distortion. However, a practical but tough problem is how small the change of metrics will lead to a satisfactory deformation. Moreover, this strategy of adding small changes of metrics converges very slowly. In summary, the major difficulty for generating a deformation with enforced metrics is how to efficiently and effectively obtain an initial guess to be further enhanced by an optimization with hard constraints, which is tackled in this paper.

1.1. Main Results

We solve the problem of metrics-enforced deformation with a framework consisting of two phases (see the work flow shown in Fig.2). In the first phase of deformation, a novel scale-driven approach is investigated to deform the input model towards the demanded values on different metrics, including length, area and volume. Shape similarity of models before and after deformation is well preserved. Our formulation leads to fast convergency in the scale-driven deformation. The measurements of specified metrics on the deformed model can be very close to the demanded values after a few iterations. In the second phase of deformation, the metrics on deformed models are further enforced in a constrained optimization with demanded metrics as hard constraints. Demanded metrics can be achieved with higher accuracy in our deformation framework. The constraints on metrics can be flexibly specified in a local or global manner according to different applications.

In the conceptual aspect, the scale factors of triangles define a configuration space and each constraint of length, area and volume defines a hyper-surface in the configuration space. Points on the intersection of these hyper-surfaces give configurations which guarantee that all constraints are satisfied. In the scale-driven step (i.e., the first phase of deformation), all triangles satisfy a conformal deformation as they are considered independently. Since all the triangles need to be blended together in the global step to generate a new mesh, least distortion will happen when the difference of two adjacent triangles' scale-factors is minimized. This idea motivates us to develop the scale-driven deformation in this paper. In each step, we project the current status onto the intersection of hyper-surfaces. After finding the optimal point projected, we use it as the final configuration to drive the blending process.

1.2. Related Work

In literature, there are a large amount of techniques for the deformation and manipulation of mesh models. This review does not aim for completeness, but provides an overview of the scope of techniques that our work relates to.

Eigensatz et al. [ESP08] introduced a curvature-domain shape processing framework that provides direct access to surface curvature. Their subsequent work in [EP09] allows direct manipulation or preservation of positional, metric, and curvature constraints on the surface of a model. In this approach, the user-specified requirements on length, area and position are converted into linear or non-linear constraints, and a global optimization functional is employed to find the deformed surface that best satisfies the constraints while preserving the details of the original mesh as much as possible. However, it cannot accurately enforce the metrics on the results of deformation. The work of spin transformations [CPS11] provides a framework for constructing conformal deformations by specifying the densities of curvatures. However, it is not clear how to further extend their work to generate deformation results with accurate values on length, area and volume. And our formulation on trying to preserve the shape similarity during deformation is in a simpler form. We consider our work as a conjunction of those approaches providing curvature control. Wang et al. [WLT12] reconstruct the vertex coordinates for a surface mesh from given edge lengths and dihedral angles. Again, the function for accurately controlling the metrics of length, area and volume is not available in their framework. The similar problem exists in other techniques trying to preserve the local shape during deformation [PDK07, KG08].

Many deformation methods based on a local/global optimization strategy have been proposed in literature. By building correspondence between the source and the target, Sumner and Popović transferred the deformation of a source mesh onto a different target mesh in [SP04]. The technique is further extended to transfer garment design in [BSBC12]. Sorkine an Alexa tried to preserve the rigidity of local transformations during deformation by a local/global approach in [SA07]. Recently, Bouaziz et al. [BDS*12] introduced a method to generate constrained deformation by defining a series of projection operators for different types of constraints and then minimizing a proximity function. Our approach decouples the deformation into two steps: 1) scale factor optimization and 2) shape optimization. This is analogous to the local and global steps in above approaches.

In another thread of research, surface deformation algorithms based on differential coordinates (ref. [SCOL*04, LSCO*04, YZX*04, LSLCO05]) focus on preserving or editing the local properties and details during deformation. Users can manipulate the derived properties such as mesh gradients or local coordinates and construct the deformed mesh by solving linear equation systems. Sheffer and Kraevoy [SK04] captured the local shape of the mesh around each vertex by building pyramid coordinates and maintain the local details under various editing operations. In our framework of deformation, the local shape is preserved by adding a mean-curvature preservation term to work together with the term for the rigidity of local frames.

Sketch and handle based deformation techniques have become popular as they provide intuitive tools for mesh editing. The work of Nealen et al. [NISA07] allows users to draw and modify control curves on the model as curve constraints and finds the surface by an optimization process. Wire based deformation methods (e.g., [SF98, GSMCO09]) exploit wire curves to deform models. As wires provide a coarse geometric representation of an object, people can intuitively edit a mesh by modifying the wire curves. An interactive deformation method in [VFTS06] deforms models with the help of vector field. When the vector field is divergence-free, the volume of a model is preserved during the deformation. However, these approaches cannot provide complex metrics-enhancement as ours.

The problem of unwanted distortion in constrained mesh deformation is also tackled by Yang et al. in [YYPM11]. They formulate the constrained mesh deformation in high dimensional spaces. The intersection of all hyperplanes determined by the functions of constraints gives the surfaces that satisfy the prescribed constraints. Then, the deformed mesh can be found by adding fairness and other requirements of surface quality. Recently, the idea is further extended in [DBD*13] to explore local modifications of constrained meshes. Although the framework of constrained mesh deformation introduced in these approaches is general, we dedicate to solve the mesh deformation problems that are driven by user-specified metrics (i.e., length, area and volume) and solve this particular type of problems by a simpler formulation.

1.3. Common Notations

An input model $\mathcal{M}_s$ is represented by a triangular mesh $\mathcal{M}_s = (\mathcal{V}; \mathcal{E}; \mathcal{F})$, where $\mathcal{V} = \{v_i\}$, $\mathcal{E} = \{e_{ij}\}$ and $\mathcal{F} = \{f_{ijk}\}$ denote the sets of vertices, edges and faces respectively. The position of a vertex v_i is given by $\mathbf{v}_i \in \Re^3$. For all constraints in the set, $\mathcal{C}$, they are classified into limited k categories: $\mathcal{C}_1$, $\mathcal{C}_2$, …, $\mathcal{C}_k$. Moreover, all values from the original model are denoted with a superscript 'o' and those computed from current status are represented with a superscript 'c'.

2. Scale-Driven Deformation

To formulate the scale-driven deformation, we first introduce how to construct local frames for an input mesh surface. After that, a method is presented to estimate the optimal scale factors for all local frames to approach the demanded values on user-specified metrics. The estimated scale factors will be used to drive the subsequent step of shape preserved deformation. Iteratively applying these two steps of scale factor

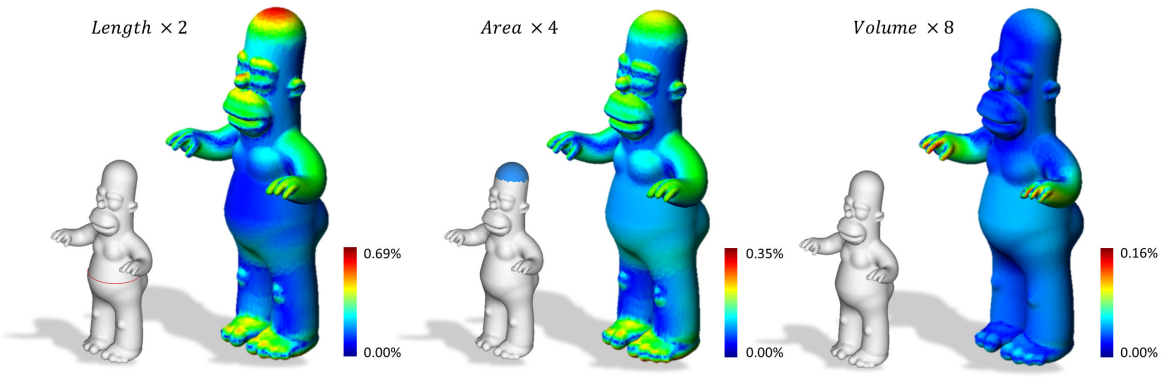

Figure 3: *The input model is deformed by specifying demand on different metrics, and the ideal results should be the same as scaling the model by* 2.0*. The publicly available PolyMeCo [Sil08] is employed to analyze and visualize the geometric error (w.r.t. the diagonal length of bounding box). Only very small shape approximation errors are generated on our results. In other words, the shape of input model is well-preserved by our scale-driven deformation framework.*

estimation and shape-preserved deformation can deform an input mesh surface gradually to meet the requested values on a variety of metrics.

2.1. Local Frame Structure

The requirement of shape preservation in mesh deformation can be converted into keeping invariant normals on faces (or invariant relative orientation between adjacent faces). However, a formulation with more stable computation is based on constructing a local frame structure (ref. [SP04]). For a triangle t with three vertices, $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$, the forth artificial vertex $\mathbf{v}_4$ is added by offsetting $\mathbf{v}_1$ along the face normal as

$$\mathbf{v}_4 = \mathbf{v}_1 + \frac{(\mathbf{v}_2-\mathbf{v}_1)\times(\mathbf{v}_3-\mathbf{v}_2)}{\sqrt{\|(\mathbf{v}_2-\mathbf{v}_1)\times(\mathbf{v}_3-\mathbf{v}_2)\|}}.$$

With the help of this artificial vertex, a local frame tensor can be constructed as $[\mathbf{v}_2-\mathbf{v}_1, \mathbf{v}_3-\mathbf{v}_1, \mathbf{v}_4-\mathbf{v}_1]$. This formulation integrates the shape and the orientation of triangles into a single formula. Thus, the shape preservation can be realized by the conservation of local frames.

To provide flexibility, we add a scale factor s to every local frame. Specifically, the local frame tensor of a triangle face can be represented as

$$\mathbf{V} = [s(\mathbf{v}_2-\mathbf{v}_1), s(\mathbf{v}_3-\mathbf{v}_1), s(\mathbf{v}_4-\mathbf{v}_1)]. \quad (1)$$

With the help of this tensor, our deformation framework can promptly approach user-specified values on different metrics while preserving the shape of an input model (see Fig.3).

2.2. Scale Factor Estimation

In the step of scale factor estimation, we attempt to find the best values of the scale factors to fit the demanded metrics. It is assumed that all constraints can be expressed as functions

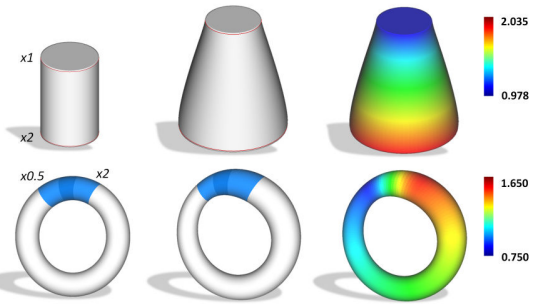

Figure 4: *Estimation of scale factors: (left) the input with demands on multiple metrics, (middle) shape obtained after a few iterations of scale-driven deformation, and (right) distribution of estimated scale factors.*

of scale factors (e.g., $C_i(s_1, s_2, \cdots, s_n)$ with n being the number of triangles on an input model). The formulations of the metrics-based constraints in the form of scale factors will be detailed in Section 3.

To achieve the goal of shape preservation, the difference of scale factors on two adjacent faces should be minimized. Meanwhile, the metrics-based constraints must be satisfied. This can be formulated as

$$\begin{aligned} &\min_{s_1,s_2,\cdots,s_n} \sum_{(i,j)\in\mathcal{Q}} (s_i - s_j)^2 \\ &s.t. \quad C_i = 0, \quad i = 1,\cdots,k. \end{aligned} \quad (2)$$

where $\mathcal{Q}$ is the set of pairs of adjacent faces. However, the Hessian matrix of this formulation is not full rank. A regu-

larization term is added to resolve this problem.

$$\min_{s_1,s_2,\cdots,s_n} k_d \sum_{(i,j)\in\mathcal{Q}} (s_i - s_j)^2 + k_r \sum_{i=1}^{n} (s_i - 1)^2 \qquad (3)$$
$$s.t. \quad C_i = 0, \quad i = 1,\cdots,k.$$

with $k_d = 1.0$ and $k_r = 0.01$. The scale factors on triangles can then be determined by using Newton's approach. Figure 4 shows the scale factors determined by this estimation method when different demanded metrics are specified.

2.3. Shape Preservation

After determining the optimal scale factors on all triangular faces, we need to change the positions of vertices to let a model deformed according to the scale factors. One has to notice that the shapes of neighboring triangles may not be compatible with each other when they are scaled exactly according to the scale factors. The frames on two faces are compatible only when their scale factors are exactly the same. Thus we should find a global balance to the configuration of scale factors while preserving local frames as much as possible.

Considering a local frame tensor $\mathbf{V}$ and its variant $\mathbf{V}'$, an affine transformation matrix is defined as $\mathbf{V}'\mathbf{V}^{-1}$ to evaluate the difference between its two states ($\mathbf{V}$ and $\mathbf{V}'$) in ignorance of translation. The deformation transfer framework in [SP04] employs an optimization to let this affine transformation be close to a target transformation, $\mathbf{T}$, by minimizing the energy, $\sum \|\mathbf{V}'\mathbf{V}^{-1} - \mathbf{T}\|_F^2$. Here, in our formulation, the new local frame should be close to the scaled old one (with the scale factor, s, determined in the above step). Therefore, we conduct the following frame-based energy functional for shape optimization

$$E_f = \sum_{i=1}^{n} w_i \|\mathbf{V}_i - s_i \mathbf{V}_i^c\|_F^2, \qquad (4)$$

where $\mathbf{V}_i$ is the i-th unknown new frame, $\mathbf{V}_i^c$ is the current i-th frame, and $\|\cdots\|_F$ is the Frobenius norm for matrices. This functional actually minimizes an energy similar to $\sum \|\mathbf{V}'\mathbf{V}^{-1} - s\mathbf{I}\|_F^2$. For a triangular mesh surface, the weight w_i is chosen as $A_i^o / \sum_i A_i^o$ (with A_i^o being the area of i-th triangle before deformation) to balance the effect of triangles with different areas.

The frame-based term does not place constraints on remaining the distribution of triangles in the local span of a vertex, therefore only using the frame-based term cannot preserve the local smoothness of a deformed model (see the upper-right of Fig.5). An additional mean curvature based energy [MDSB02] is added for this purpose.

$$E_m = \sum_{i\in\mathcal{V}} \| \sum_{j\in\mathcal{N}(i)} \omega_{ij} \left((\mathbf{v}_i - \mathbf{v}_j) - (\mathbf{v}_i^c - \mathbf{v}_j^c) \right) \|^2, \qquad (5)$$

where $\mathcal{N}(i)$ denotes the 1-ring neighbor vertices of $\mathbf{v}_i$ and ω_{ij}s are the cotangent weights computed from the input

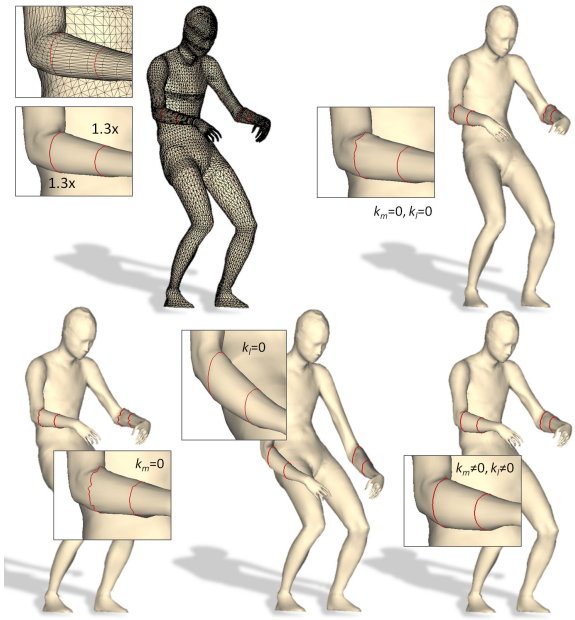

Figure 5: *Different deformation results by different combination of weights: k_f, k_m and k_l. Note that, $k_f = 1000.0$ is used in all tests.*

mesh. However, this energy is not strong enough to control the relative orientation between neighboring local frames. A regularization term based on mean value Laplacian [Flo03] is added. For a vertex, $\mathbf{v}_i$, its mean value Laplacian, $L(\mathbf{v}_i)$, is expected to be invariant with the one on the current position, $L(\mathbf{v}_i^c)$. However, as Laplacian is scale-dependent, we formulate a term preserving the direction (but not the value) of Laplacian, which is

$$E_l = \sum_{i\in\mathcal{V}} \|L(\mathbf{v}_i) \times L(\mathbf{v}_i^c)\|^2. \qquad (6)$$

By integrating all the energy terms, the shape optimization is formulated as

$$\min_{\mathbf{v}_1,\mathbf{v}_2,\cdots,\mathbf{v}_m} k_f E_f + k_m E_m + k_l E_l. \qquad (7)$$

Generally, the weights are set as $k_f = 1000.0$ and $k_m = k_l = 1.0$ to balance the contribution of different terms. The results of applying different combinations on the values of k_f, k_m and k_l are illustrated in Fig.5.

2.4. Iterative Deformation and Metrics Enforcement

Applying only one step of scale factor estimation followed by another step of shape optimization cannot obtain a satisfactory result when multiple metrics are demanded. An iterative process is needed to repeatedly apply these two steps to let the deformation converge to demanded values on those user-specified metrics. In each iteration, we first estimate the

optimal scale factors on the current mesh. Then, local frames are constructed on the current mesh and the shape optimization step is applied to deform the mesh surface. The iteration is terminated when the maximal change of scale factors on triangular faces is less than a threshold (e.g., 0.05).

Finally, the mesh surface is fine-tuned to enforce the metrics (formulated as hard constraints, F_i) by

$$\begin{aligned} &\min_{\mathbf{v}_1,\mathbf{v}_2,\cdots,\mathbf{v}_m} k_f E_f + k_m E_m + k_l E_l \\ &s.t. \quad F_i = 0, \quad \text{for} \quad i = 1,\cdots,k. \end{aligned} \tag{8}$$

which can be solved by using Newton's approach. As the Newton's approach may lead to high distortion (ref. [BS08]), we emphasize the terms of shape smoothness in this fine-tuning step – i.e., k_f, k_l and k_m are set to 1, 100 and 100. As the objective functions defined in Eqs.(3), (7) and (8) all have the quadratic form, the numerical computation is stable and converges very fast. The minimization of Eq.(8) finally yields a shape-preserved model that satisfies all hard constraints.

3. Constraints for Metrics

In this section, we formulate the constraints of *Length*, *Area*, and *Volume* to fit our optimization framework.

3.1. Length Constraint

The demand on length is allowed to be specified on curves in our framework. We represent a curve on the mesh surface as a set of connected line segments and each segment belongs to a triangular face. The line segment in a triangle can be uniquely defined by two characteristic nodes, the positions of which are encoded with the triangle's vertices by the barycentric coordinates. While fixing the barycentric coordinates, line segments can be deformed together with the mesh surface. As a result, the length of a line segment can be evaluated by the positions of its characteristic nodes, which are linear combination of mesh vertices.

In the step of scale factor estimation, the length of one segment can be expressed by its current length on a face and its corresponding scale factor. Therefore, we define the length of a curve on the mesh surface $\mathcal{M}_s$ as $L_{\mathcal{L}} = \sum_{i\in\mathcal{F}_{\mathcal{L}}} s_i l_i^c$, where l_i^c indicates the current length of i-th line segment on the curve $\mathcal{L}$ and $\mathcal{F}_{\mathcal{L}}$ is the index set of triangles containing the line segments of $\mathcal{L}$. Therefore, the length constraint for the step of scale factor estimation is defined as

$$C_{length} = \sum_{\mathcal{L}} \left[\frac{1}{\hat{L}_{target}} \left(\sum_{i\in\mathcal{F}_{\mathcal{L}}} s_i l_i^c \right) - 1 \right]^2 \tag{9}$$

with $\hat{L}_{target}$ being the target length of the curve $\mathcal{L}$. While in the fine-tuning step of the shape optimization, the hard constraints for length enforcement become

$$F_{length} = \sum_{\mathcal{L}} \left[\frac{1}{\hat{L}_{target}} \left(\sum_{i\in\mathcal{F}_{\mathcal{L}}} l_i \right) - 1 \right]^2. \tag{10}$$

Note that, in this formulation, l_i is represented by the linear combination of the positions of mesh vertices with the help of barycentric coordinates.

3.2. Area Constraint

The area of a region $\mathcal{P}$ on the input mesh $\mathcal{M}_s$ is the total area of triangles fall in this region. Users can define several such regions on $\mathcal{M}_s$ and specify the target values of their area to drive a deformation (see Figs.4 and 6 for examples). The constraint for areas in the step of scale factor estimation is defined as

$$C_{area} = \sum_{\mathcal{P}} \left[\frac{1}{\hat{A}_{target}} \left(\sum_{f\in\mathcal{F}_{\mathcal{P}}} s_f^2 a_f^c \right) - 1 \right]^2 \tag{11}$$

in terms of the scale factors, s_f, to be determined. In this formulation, a_f^c is the current area of a face f and $\mathcal{F}_{\mathcal{P}}$ is the set of triangles in a user-specified region, $\mathcal{P}$.

When taking the fine-tuning shape optimization, the hard constraint on regions with demanded areas is defined as

$$F_{area} = \sum_{\mathcal{P}} \left[\frac{1}{\hat{A}_{target}} \left(\sum_{f\in\mathcal{F}_{\mathcal{P}}} a_f \right) - 1 \right]^2, \tag{12}$$

where the area a_f of triangle f is $\frac{1}{2}\|(\mathbf{v}_2^f - \mathbf{v}_1^f) \times (\mathbf{v}_3^f - \mathbf{v}_1^f)\|$ in terms of its three vertices $\mathbf{v}_1^f$, $\mathbf{v}_2^f$ and $\mathbf{v}_3^f$.

3.3. Volume Constraint

The volume of a surface mesh can be represented as the sum of signed volume of virtual tetrahedrons. The virtual tetrahedron t for a triangle face f (with vertices $\mathbf{v}_1^f$, $\mathbf{v}_2^f$ and $\mathbf{v}_3^f$) can be constructed by introducing origin as the fourth vertex, so that its volume is $v_t = \frac{1}{6}\mathbf{v}_1^f \cdot ((\mathbf{v}_2^f - \mathbf{v}_1^f) \times (\mathbf{v}_3^f - \mathbf{v}_2^f))$.

When estimating the scale factors, the scaled volume of a virtual tetrahedron t is $s_t^3 v_t$. Again, the constraint used in generating optimal shape factors is

$$C_{volume} = \sum_{\mathcal{V}} \left[\frac{1}{\hat{V}_{target}} \left(\sum_{t\in\mathcal{F}_{\mathcal{V}}} s_t^3 v_t^c \right) - 1 \right]^2 \tag{13}$$

with v_t^c being the current volume of a virtual tetrahedron t and $\mathcal{F}_{\mathcal{V}}$ being the set of constrained tetrahedra.

In the fine-tuning step to enforce constraint on volume, the constraint is formulated as

$$F_{volume} = \sum_{\mathcal{V}} \left[\frac{1}{\hat{V}_{target}} \left(\sum_{t\in\mathcal{F}_{\mathcal{V}}} v_t \right) - 1 \right]^2 \tag{14}$$

again in terms of mesh vertices.

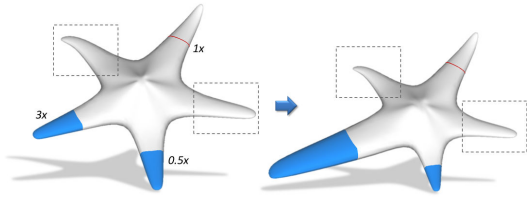

Figure 6: *The starfish is deformed with two area constraints while the volume is preserved.*

Global volume preservation is one of most common requirements in model design (see Fig.6 for an example). After a simple extension, our formulation can deal with both global and local volume constraints. To add local volume constraints, we only need to take an additional tessellation as a preprocessing step to make a locally closed volume.

4. Details of Numerical Scheme

The deformation framework proposed in this paper can be implemented by solving three optimization problems. The shape optimization step (Eq.(7)) is a least-square problem that can be efficiently solved. The steps of shape factor estimation (Eq.(3)) and fine-tuning (Eq.(8)) are formulated as constrained optimization, which is converted into an augmented objective function with Lagrange multiplier as

$$J(\mathbf{x},\lambda) = E(\mathbf{x}) + \sum_i \lambda_i C_i(\mathbf{x}).$$

The augmented objective function can be solved by using Newton's approach [MNT04]. The constraints are classified into k categories with k Lagrange multipliers $\lambda_1, \lambda_2, \cdots, \lambda_k$, where the constraints in the same type are grouped into the same category. Specifically, we have three categories – *length*, *area* and *volume*. The reason why we classify the constraints rather than combine all constraints together is that different types of constraints have different speed of convergence in terms of scale factors (i.e., linear, quadratic and cubic orders). The Lagrange multipliers $\lambda_1, \lambda_2, \cdots, \lambda_k$ determined in every iteration step automatically adjust the weights for each type of constraints. This formulation helps avoid the interference among the different types of constraints and significantly improves the convergence speed of computation.

Motivated by [SLMB05], we can further simplify the computational complexity by the sequential linearly constrained programming. Specifically, the second order derivatives of the constraints in the Hessian matrix $\nabla^2 J$ are neglected. The linear system to be solved in each step of the iteration can be expressed as

$$\begin{bmatrix} \mathbf{H} & \mathbf{L}^T \\ \mathbf{L} & 0 \end{bmatrix} \begin{bmatrix} \delta_{\mathbf{x}} \\ \delta_\lambda \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\mathbf{x}} \\ \mathbf{b}_\lambda \end{bmatrix}, \tag{15}$$

where $\mathbf{H} = \nabla^2 E(\mathbf{x})$ is the Hessian matrix of $E(\mathbf{x})$ and $\mathbf{L} = \frac{\partial}{\partial \lambda} \nabla C(\mathbf{x})$. The vectors on the right-hand side are

$$\mathbf{b}_{\mathbf{x}} = -\nabla E(\mathbf{x}) - \sum_i \lambda_i \nabla C_i(\mathbf{x}) \text{ and } \mathbf{b}_\lambda = \{-C_i(\mathbf{x})\}.$$

The linear system in Eq.(15) can be further converted into

$$\delta_{\mathbf{x}} + \mathbf{H}^{-1}\mathbf{L}^T\delta_\lambda = \mathbf{H}^{-1}\mathbf{b}_{\mathbf{x}} \tag{16}$$

$$\mathbf{L}\delta_{\mathbf{x}} = \mathbf{b}_\lambda. \tag{17}$$

Then, the value of δ_λ can be first determined by eliminating the above two equations to

$$\mathbf{L}\mathbf{H}^{-1}\mathbf{L}^T\delta_\lambda = \mathbf{L}\mathbf{H}^{-1}\mathbf{b}_{\mathbf{x}} - \mathbf{b}_\lambda. \tag{18}$$

After solving Eq.(18), the unknown $\delta_{\mathbf{x}}$ can be obtained by substituting the value of δ_λ into Eq.(16). During the computation, we need to calculate $\mathbf{H}^{-1}\mathbf{b}_{\mathbf{x}}$ and $\mathbf{H}^{-1}\mathbf{L}^T$, which can be obtained by solving

$$\mathbf{H}\mathbf{y} = \mathbf{b}_{\mathbf{x}} \text{ and } \mathbf{H}\mathbf{Y} = \mathbf{L}^T,$$

where $\mathbf{y}$ is a vector with $size(\mathbf{x})$ components and $\mathbf{Y}$ is a $size(\mathbf{x}) \times k$ matrix. As a result, $\mathbf{H}^{-1}\mathbf{b}_{\mathbf{x}} = \mathbf{y}$, $\mathbf{H}^{-1}\mathbf{L}^T = \mathbf{Y}$, and Eq.(18) becomes

$$\mathbf{L}\mathbf{Y}\delta_\lambda = \mathbf{L}\mathbf{y} - \mathbf{b}_\lambda.$$

When $k = 1$, the value of δ_λ can be directly determined. For cases with multiple types of constraints (i.e., $k \geq 2$), *Singular Value Decomposition* (SVD) is applied to improve the robustness of solving the unknowns, δ_λ.

In our scale-driven deformation, the Hessian matrices $\mathbf{H}$ of objective functions in Eqs.(3) and (8) are fixed during the iterations of solving constrained optimization. Therefore, we factorize $\mathbf{H}$ into two triangular matrices by the LU-decomposition [LDG], then all systems of linear equations using $\mathbf{H}$ as coefficient matrix can be solved by applying two back substitutions of the lower and upper triangular matrices [PTVF95]. In summary, the LU-decomposition of Hessian matrix $\mathbf{H}$ is taken only once during the iterations of solving constrained optimization. All the rest computation is based on back-substitution and matrix multiplication, which can be efficiently performed. To guarantee the convergence of hard constraints, we also apply a linear search scored by the squared sum of constraints as $(\sum_i \lambda_i C_i(\mathbf{x}))^2$ to determine an optimal updating scale after solving $\delta_{\mathbf{x}}$ in each iteration of the Newton's approach.

5. Results

5.1. Validation and Comparison

We have tried our scale-driven deformation on a variety of models using different combinations of metrics as input. The results are encouraging. Figures 1 and 2 show the deformation driven by demanded lengths on user-specified curves. For the deformations with single enforced metric, the shape-approximation errors between the results generated by our approach and the ground truth have been analyzed in Fig.3. Our approach can produce very accurate conformal deformation. When applying multiple demanded metrics as input (e.g., the starfish example shown in Fig.6), our framework

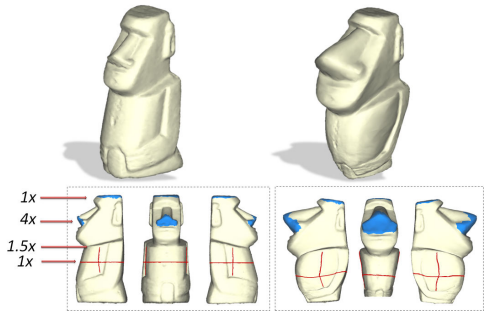

Figure 7: *Volume preserved manipulation of the Moai model with multiple requests on length and area.*

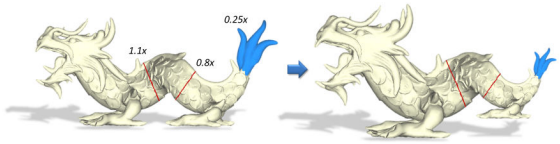

Figure 8: *Volume preserved deformation is applied to the dragon model with multiple requests on length and area.*

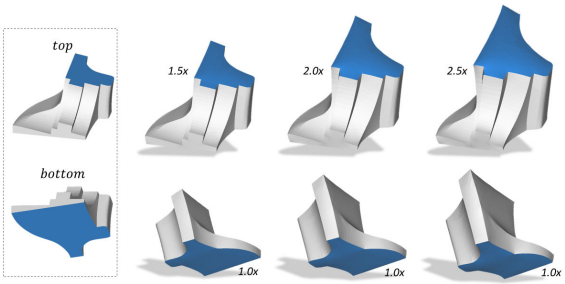

Figure 9: *Area of the fandisk model at the bottom is fixed while the area on the top is increased by* 50%*,* 100% *and* 150% *respectively. Sharp features are well preserved on the deformation results.*

Table 1: *Statistics of Metrics on Resultant Models*

Model	Starfish (Fig.6)			
Metrics	Original	Target	Result	Error (%)
Area	55.45	499.08	499.08	0.001
Area	107.75	107.75	107.76	0.009
Model	Moai (Fig.7)			
Metrics	Original	Target	Result	Error (%)
Length	14.23	14.23	14.31	0.56
Length	2.56	3.83	3.82	0.26
Length	2.35	3.52	3.51	0.34
Area	5.83	5.82	5.82	0.03
Area	2.26	9.05	9.02	0.27
Volume	117.99	117.99	118.72	0.62
Model	Dragon (Fig.8)			
Type	Original	Target	Result	Error (%)
Area	1829.79	457.44	457.42	0.005
Length	113.99	91.19	91.28	0.095
Length	106.53	117.18	117.01	0.149
Volume	154,328	154,328	154,418	0.058

can automatically compromise among the different types of metrics. Three examples with more complex shape are shown in Figs.7, 8 and 9. The geometry details and sharp features are well-preserved by our approach.

Measurements are taken on resultant models to check whether the demanded values have been achieved on the user-specified metrics. Statistics are listed in Table 1. Benefited by the new formulation of scale-driven deformation, metrics are well enforced in our framework – all examples show less than 1% of error on different types of metrics. Our implementation of [EP09] cannot generate results with such strictly preserved metrics – see the comparison shown in Fig.10. When assigning demanded metrics as hard constraints, highly distorted models are generated. Progressively changing the values on demanded metrics can somewhat reduce the distortion. However, deformations generated by conducting this strategy still cannot produce results as good as ours after taking a time-consuming procedure of progressive enforcement. A case study has been shown in Fig.1.

Another comparison is taken by trying to realize the functions offered in this paper under the framework of Shape-Up [BDS*12], which is a local/global approach for constrained geometry processing. For the example shown in Fig.11, we can obtain the local shape of every triangle by scaling if it belongs to the selected top region (i.e., the blue ones). For those triangles not falling in the blue region, their current shape is used as the local shape. Then, the global blending will assemble all the local shape of triangles into a global shape of the mesh surface. The result of Shape-Up is as shown in Fig.11(d). Compared with our result (Fig.11(c)), their approach generates a result with worse surface quality. Moreover, it is too restrictive to scale each triangle by $\sqrt{2}$ while the metric is only specified as making the total area of these triangles double. More seriously, it will have difficulty to process intersected demands on metrics (e.g., the example shown at the bottom of Fig.4 and even more complex combination of multiple types of metrics).

5.2. Position and Curvature Control

Position handles can be added into our deformation framework to work together with the demanded metrics. Specifically, Eq.(7) is modified into the following form

$$\min_{\mathbf{v}_1,\mathbf{v}_2,\cdots,\mathbf{v}_m} k_f E_f + k_m E_m + k_l E_l + k_p E_p. \qquad (19)$$

by adding a positional term

$$E_p = \sum_{\mathbf{v}_p \in \mathcal{P}} \|\mathbf{v}_p - \mathbf{v}_p^t\|^2$$

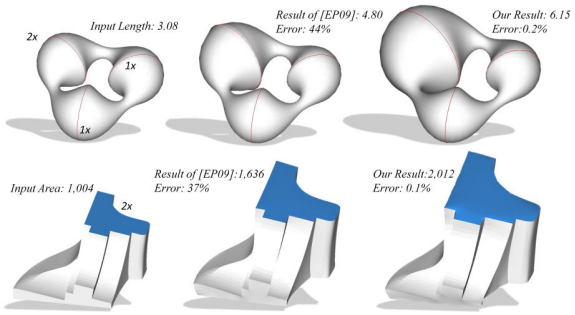

Figure 10: *Comparison with the results generated by our implementation of [EP09].*

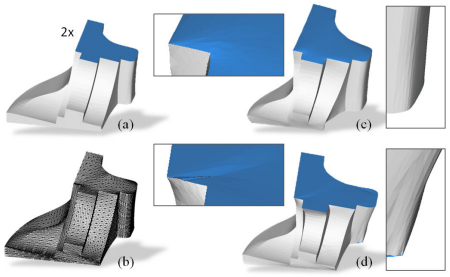

Figure 11: *Comparison with Shape-Up [BDS*12]: our approach has a better shape preservation (c) and the result of Shape-Up in (d) has irregularly distorted sharp-edges.*

with $\mathbf{v}_p \in \mathcal{P}$ being a vertex in the set of positional handles $\mathcal{P}$ and $\mathbf{v}_p^t$ being the target position of $\mathbf{v}_p$. The weight, $k_p = 1000.0$, is used in our experimental tests. Similarly, the positional term E_p is also added into Eq.(8) for the same purpose as

$$\begin{aligned} &\min_{\mathbf{v}_1,\mathbf{v}_2,\cdots,\mathbf{v}_m} k_f E_f + k_m E_m + k_l E_l + k_p E_p \\ &s.t. \quad F_i = 0, \quad \text{for} \quad i = 1,\cdots,k. \end{aligned} \tag{20}$$

As a consequence, users can control the deformation by using positional handles together with demanded metrics. An example is shown in Fig.12.

Moreover, we can also incorporate the curvature control into our scale-driven deformation framework. The relative value of mean curvature can be controlled by scaling the target vectors in Eq.(5) as follows:

$$E_m = \sum_{i\in\mathcal{V}} \| \sum_{j\in\mathcal{N}(i)} \omega_{ij} \left((\mathbf{v}_i - \mathbf{v}_j) - c_i(\mathbf{v}_i^c - \mathbf{v}_j^c) \right) \|^2, \tag{21}$$

where the scaling factor c_i is used to control the change of mean curvature vectors during the deformation. Although this is not a precise control, designers can still use this to somewhat control the shape of deformation by the relative change on curvatures – see an example in Fig.13. In fact,

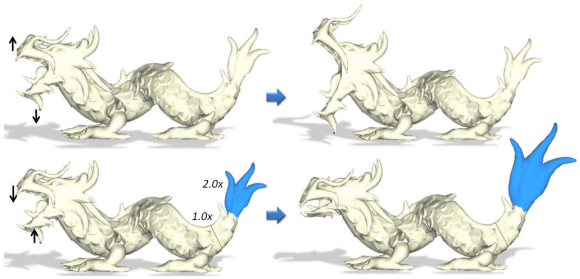

Figure 12: *Positional handles can be added into our framework of scale-driven deformation.*

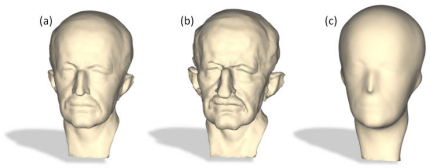

Figure 13: *Designers can use scaling factors on curvature to control the shape in our framework: (a) an input model, (b) scaling mean curvature with factor $\times 2.0$, and (c) inversely scaling the curvature by $\times 0.1$.*

for industrial designers it is also hard to ask them to input quantitative requirements on curvatures.

5.3. Application

The technique developed in this paper provides a very user-friendly interface for the applications in design. Users can easily use sketches to draw curves and select regions to specify demanded metrics on lengths and areas. For example, the Moai model shown in Fig.7 can be easily deformed to its inflated version. More operations on designing a model with the help of user-specified metrics can be found in Fig.14, where a curve is specified on the nose and two ears are selected as regions with demanded areas in design.

The second application shown here comes from an industrial project of producing personalized wetsuit that can gen-

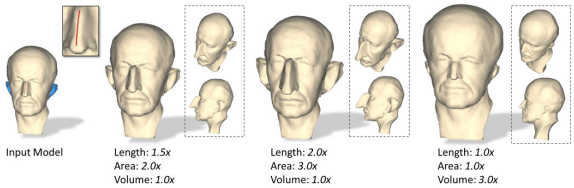

Figure 14: *Application on the design by metrics: A head model can be deformed to different variations with different demanded metrics as input.*

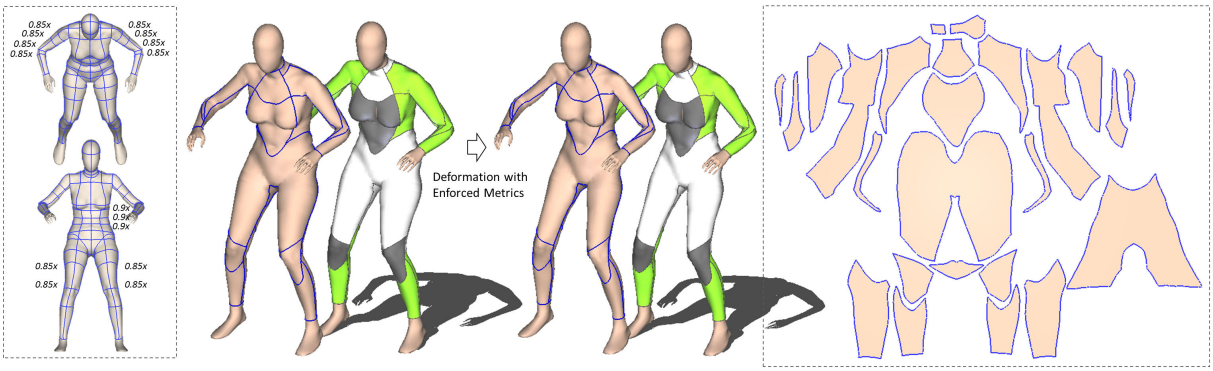

Figure 15: *Application on the design of compression wetsuit: Different demanded compression ratios (in terms of length variation on the feature curves) are specified according to the study of ergonomics. After deforming the human model by our framework according to the input length requirements, a new model with deformed wetsuit can be obtained to generate 2D patterns for compression. Patterns generated from a deformed human body is shown on the right.*

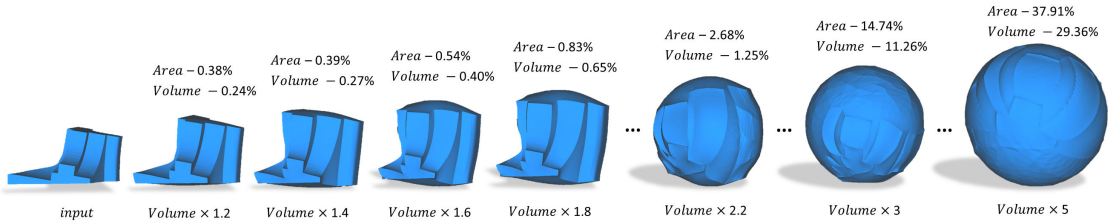

Figure 16: *Contradiction between demanded metrics: progressively increasing the demanded volume while preserving the original surface area can lead to a contradiction. As a result, the accuracy on demanded metrics drops significantly. Note that, by theoretically analyzing the area and volume of a sphere, the maximal volume with fixed area can be achieved in this example is* $2.202\times$ *of the original volume.*

erate certain compression on the human body when wearing the wetsuit (see Fig.15). With the help of advanced geometric modeling technique, a personalized wetsuit can be fabricated by 1) cutting the scanned 3D human models into smaller pieces according to the styling curves (e.g., by [LLP05]) and 2) flattening the 3D pieces into 2D patterns for the fabrication (e.g., by [LZX*08]). The new development in this industrial application requests generating controlled compression onto the surface of human body. The level of compression is controlled by the percentage of stretch on feature curves, which can be realized by our deformation framework (see the right of Fig.15). Then, the patterns of compression wetsuit can be obtained by flattening the 3D mesh pieces generated from the deformed human body.

5.4. Discussion

Although the scale-driven deformation framework proposed in this paper can successfully handle the input with demand on different types of metrics which may intersect, the demanded metrics will have large errors when the metrics are specified in the way leading to contradiction. For example, when fixing the surface area while increasing its volume tremendously till the volume to area ratio is even greater than a sphere, there exits no solution in reality. The optimization framework will automatically sacrifices the accuracy on demanded metrics to generate a model which can be presented in the Euclidean space (see Fig.16 for an example).

Scale-aware deformation is relevant to the conformality of surfaces. Eq.(3) is formulated to minimize the difference of scaling factors between neighboring faces, which is similar to the requirement on the conformality of deformation. However, in the global step for reconstructing positions of vertices, our formulation tends to preserve the geometric details. Crane et al. proposed an approach in [CPS11] to preserve the conformality of surfaces during deformation. We compare our result with the deformation generated by [CPS11] in Fig.17. The quasi-conformal error on each face is generated by the ratio of the largest to smallest singular values of Jacobian (details can be found in [SSGH01]). The ideal error is 1.0, which means only uniform scaling and rotation happen. It can be found that preserving the geometric details during deformation is different from the preservation

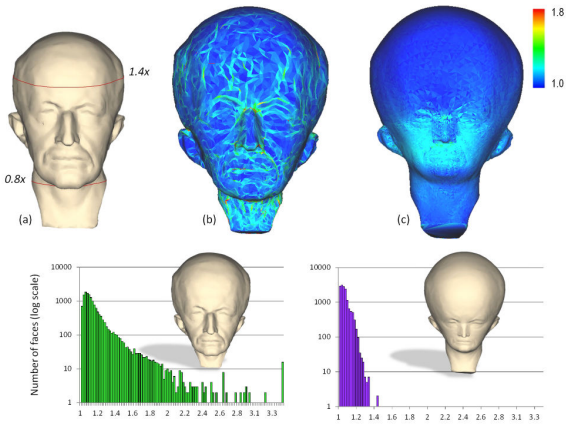

Figure 17: *The difference between our approach and [CPS11]: (a) an input model, (b) the result of our approach, (c) the result of deformation generated by the tools of [CPS11]. Unlike [CPS11] that generates a conformal deformation, our formulation tends to preserve the local geometric details. The color map and the histogram of quasi-conformal error can be generated by the method of [SSGH01], where an error of* 1.0 *in each face is ideal.*

of conformality. One of our future work is how to realize accurate control of metrics in conformal deformations.

There are two major limitations of our framework. First, the constraint energy in Eq.(3) must be a function of scale factors. Thus the constraints which could not be expressed by scale factors are currently not compatible with our framework (e.g. curvature and angle). This can be one of our future work to improve the compatibility of this framework. Second, as the barycentric coordinate is employed to encode the shape of a curve with the vertices on mesh surfaces, curves on deformed models can be highly distorted when the quality of mesh is very poor. The problem can be resolved if the curves coincident with triangular edges. Therefore, a preprocessing step of remeshing is needed when the input meshes have a lot of triangles in the 'needle' or 'cap' shape.

As the techniques of non-linear optimization are employed to formulate the deformation framework, the computing cost of this approach is much higher than the linear deformation techniques (ref. [BS08]) even after taking the pre-factorization step proposed in Section 4. For a model with around 10k faces, our approach takes $30 \sim 100$ seconds on a 2.83GHz Intel Xeon E5440 Dual Core with 4GB of memory. In practice, the time cost depends on the configuration of demanded metrics. Generally, the computation converges faster with no intersection between metrics, and takes longer time when they intersect. The relatively slow speed in computation is actually caused by setting the maximum iteration step in each optimization to 500 to get precise results. As a matter of fact, the optimization energies defined in our framework drop significantly in the first few steps – thanks to their quadratic forms. Based on this, the performance of our approach can be improved in a tricky way. Besides computing the finally converged results, we can generate a 'preview' of deformation by applying a fewer steps of iteration. For example, when applying 5 steps of iteration in all our experiments, the results of such 'preview' can be generated in a speed as follows. For models with 5k faces, the computation time is < 1 second. For models with 10k faces, the time is less than 2 seconds; and models with 50k faces, it takes less than 5 seconds. We notice that even when only 5 steps are applied, the errors of constraints on demanded metrics will commonly drop to less than 10% (see the model shown in Fig.2 as an example).

6. Conclusion

We present a scale-driven deformation framework in this paper to provide a tool for deforming 3D models by metrics. The energy functional are defined with simplicity, numerical stability and computational efficiency. As a result, demanded metrics on length, area and volume can be easily achieved. Experimental tests show that deforming a mesh with single or multiple metrics-based hard constraints under our framework can obtain high accuracy. In summary, our approach provides an effective and efficient deformation tool for a variety of applications in industry.

References

[AB97] AUBERT F., BECHMANN D.: Volume-preserving space deformation. *Comput. Graph. 21*, 5 (1997), 625–639. 2

[BDS*12] BOUAZIZ S., DEUSS M., SCHWARTZBURG Y., WEISE T., PAULY M.: Shape-Up: Shaping discrete geometry with projections. *Computer Graphics Forum 31*, 5 (2012), 1657–1667. 3, 8, 9

[BK04] BOTSCH M., KOBBELT L.: An intuitive framework for real-time freeform modeling. *ACM Trans. Graph. 23*, 3 (2004), 630–634. 1

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Trans. Vis. and Comp. Graph. 14*, 1 (2008), 213–230. 1, 2, 6, 11

[BSBC12] BROUET R., SHEFFER A., BOISSIEUX L., CANI M.-P.: Design preserving garment transfer. *ACM Trans. Graph. 31*, 4 (2012), 36. 3

[CPS11] CRANE K., PINKALL U., SCHRÖDER P.: Spin transformations of discrete surfaces. *ACM Trans. Graph. 30*, 4 (2011), 104:1–104:10. 3, 10, 11

[DBD*13] DENG B., BOUAZIZ S., DEUSS M., ZHANG J., SCHWARTZBURG Y., PAULY M.: Exploring local modifications for constrained meshes. *Comput. Graph. Forum 32*, 2 (2013), 11–20. 3

[EP09] EIGENSATZ M., PAULY M.: Positional, metric, and curvature control for constraint-based surface deformation. *Computer Graphics Forum 28*, 2 (2009), 551–558. 1, 3, 8, 9

[ESP08] EIGENSATZ M., SUMNER R. W., PAULY M.: Curvature-domain shape processing. *Computer Graphics Forum 27*, 2 (2008), 241–250. 3

[Flo03] FLOATER M. S.: Mean value coordinates. *Computer aided geometric design 20*, 1 (2003), 19–27. 5

[GSMCO09] GAL R., SORKINE O., MITRA N. J., COHEN-OR D.: iWIRES: an analyze-and-edit approach to shape manipulation. *ACM Trans. Graph. 28*, 3 (2009), 33:1–33:10. 3

[HML00] HIROTA G., MAHESHWARI R., LIN M. C.: Fast volume-preserving free-form deformation using multi-level optimization. *Computer-Aided Design 32*, 8 (2000), 499–512. 2

[KG08] KIRCHER S., GARLAND M.: Free-form motion processing. *ACM Trans. Graph. 27*, 2 (2008), 12:1–12:13. 3

[LDG] LI X. S., DEMMEL J. W., GILBERT J. R.: *The superLU library*. http://crd.lbl.gov/˜xiaoye. 7

[LLP05] LI W.-C., LEVY B., PAUL J.-C.: Mesh editing with an embedded network of curves. In *Proceedings of SMI 2005* (2005), pp. 62–71. 10

[LSCO*04] LIPMAN Y., SORKINE O., COHEN-OR D., LEVIN D., ROSSI C., SEIDEL H.-P.: Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling Applications* (2004), pp. 181–190. 1, 3

[LSLCO05] LIPMAN Y., SORKINE O., LEVIN D., COHEN-OR D.: Linear rotation-invariant coordinates for meshes. *ACM Transactions on Graphics (TOG) 24*, 3 (2005), 479–487. 1, 3

[LZX*08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S. J.: A local/global approach to mesh parameterization. In *Proceedings of the Symposium on Geometry Processing* (2008), pp. 1495–1504. 10

[MDSB02] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III* (2002), pp. 113–134. 5

[MNT04] MADSEN K., NIELSEN H. B., TINGLEFF O.: Optimization with constraints (2nd ed.). 7

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fibermesh: designing freeform surfaces with 3d curves. *ACM Trans. Graph. 26*, 3 (2007). 3

[PDK07] PARIES N., DEGENER P., KLEIN R.: Simple and efficient mesh editing with consistent local frames. In *Proceedings of Pacific Graphics 2007* (2007), pp. 461–464. 3

[PTVF95] PRESS W., TEUKOLSKY S., VETTERLING W., FLANNERY B.: Numerical recipes in C: the art of scientifc computing (2nd ed.), 1995. 7

[RSB96] RAPPOPORT A., SHEFFER A., BERCOVIER M.: Volume-preserving free-form solids. *IEEE Transactions on Visualization and Computer Graphics 2*, 1 (1996), 19–27. 2

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proceedings of the fifth Eurographics symposium on Geometry processing* (2007), pp. 109–116. 3

[SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2004), pp. 175–184. 3

[SF98] SINGH K., FIUME E.: Wires: a geometric deformation technique. In *Proceedings of SIGGRAPH 1998* (1998), pp. 405–414. 3

[Sil08] SILVA S.: *PolyMeCo: Polygonal Mesh Analysis and Comparison Tool*. http://www.ieeta.pt/polymeco/, 2008. 4

[SK04] SHEFFER A., KRAEVOY V.: Pyramid coordinates for morphing and deformation. In *Proceedings of 3D Data Processing, Visualization and Transmission* (2004), pp. 68–75. 3

[SLMB05] SHEFFER A., LÉVY B., MOGILNITSKY M., BOGOMYAKOV A.: ABF++: fast and robust angle based flattening. *ACM Trans. Graph. 24*, 2 (2005), 311–330. 7

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM Trans. Graph. 23*, 3 (2004), 399–405. 3, 4, 5

[SSGH01] SANDER P. V., SNYDER J., GORTLER S. J., HOPPE H.: Texture mapping progressive meshes. In *Proceedings of SIGGRAPH 2001* (2001), pp. 409–416. 10, 11

[VFTS06] VON FUNCK W., THEISEL H., SEIDEL H.-P.: Vector field based shape deformations. *ACM Trans. Graph. 25*, 3 (2006), 1118–1125. 1, 2, 3

[WLT12] WANG Y., LIU B., TONG Y.: Linear surface reconstruction from discrete fundamental forms on triangle meshes. *Comp. Graph. Forum 31*, 8 (2012), 2277–2287. 3

[YYPM11] YANG Y.-L., YANG Y.-J., POTTMANN H., MITRA N. J.: Shape space exploration of constrained meshes. *ACM Trans. Graph. 30*, 6 (2011), 124:1–124:12. 3

[YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.-Y.: Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph. 23*, 3 (2004), 644–651. 3