Computer-Aided Design 45 (2013) 545–550

Contents lists available at SciVerse ScienceDirect

Computer-Aided Design

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

Technical note

Hierarchical feature subspace for structure-preserving deformation

Shengfa Wang [a,b,*], Tingbo Hou [a], Shuai Li [a], Zhixun Su [b], Hong Qin [a]

[a] Department of Computer Science, Stony Brook University, USA

[b] School of Mathematic Science, Dalian University of Technology, China

ARTICLE INFO

Keywords:
Structure-preserving deformation
Feature subspace
Energy optimization
Reconstruction

ABSTRACT

This paper aims to propose a new framework for structure-preserving deformation, which is interactive, stable, and easy to use. The deformation is characterized by a nonlinear optimization problem that retains features and structures while allowing user-input external forces. The proposed framework consists of four major steps: feature analysis, ghost construction, energy optimization, and reconstruction. We employ a local structure-tensor-based feature analysis to acquire prior knowledge of the features and structures, which can be properly enforced throughout the deformation process. A ghost refers to a hierarchical feature subspace of the shape. It is constructed to control the original shape deformation in a user-transparent fashion, and speed up our algorithm while best accommodating the deformation. A feature-aware reconstruction is devised to rapidly map the deformation in the subspace back to the original space. Our user interaction is natural and friendly; far fewer point constraints and click-and-drag operations are necessary to achieve the flexible shape deformation goal. Various experiments are conducted to demonstrate the ease of manipulation and high performance of our method.

1. Introduction

The key challenge of mesh deformation is how to retain the shape functionality and visual appearance while being versatile, robust, and easy to use and supporting a wide spectrum of shape variation. Local features and global structures are two main characters of any shape. The former aims to measure local geometric saliency quantitatively, while the latter seeks to qualitatively depicts partial or global shape patterns, such as piecewise patches separated by curve network, long branches/bifurcations, or topological holes. Structure-preserving deformation especially focuses more on the global structure. An efficient click-and-drag interface that can proactively engage the user's intention and experience on how models deform always has a strong appeal to professional users and general public. This work aims to tackle these challenges, as we intend to seek as-good-as-possible approximations with lowest possible computational cost, and maximize the utility of simple click-and-drag interface.

Many existing deformation technologies typically try to preserve low-level differential properties of the edited surface that represent local details. The first-order or second-order properties of a surface [1–3], such as curvature, length, area, and local coordinates, are employed to preserve the surface details by constructing a quadratic energy. Nonetheless, they generally lack the global perception of structures, and are cumbersome to directly work with when the functional constraints are very few. More functional constraints, such as feature curves or control regions [4–6], are needed to achieve the deformation goal while preserving the features and structures. Another drawback of these methods is that directly solving the optimization problem is costly; it is heavily dependent on the size of the underlying models.

Instead of handling the surface directly, the ambient space that surrounds the mesh is used to implicitly edit the surface [7–9]. The ambient space is usually constructed by a cage or skeleton, which in turn is manipulated by the deformation scheme. The advantage of this mode is that the size of the optimization problem is reduced; this mostly depends on the ambient space rather than the resolution of the original mesh. However, certain limitations still prevail, including (but not just limited to) not being able to accommodate structure editing, lack of feature and structure awareness, complex and non-unique cage construction. Our idea is to reduce the complexity of the optimization problem by using a low-dimensional hierarchical feature subspace which maintains the features and structures of original space.

In this paper, we explore a feature-guided surface deformation, which can well preserve shape structure with very simple position constraints. By employing feature analysis based on a local structure tensor, we can learn the shape structure and geometric features of the original mesh, which are relevant throughout the shape editing session. Then, a feature subspace, called ghost, is constructed to simplify the deformation complexity and speed up

* Corresponding author at: Department of Computer Science, Stony Brook University, USA.

E-mail addresses: shengfawang@gmail.com (S. Wang), qin@cs.stonybrook.edu (H. Qin).

0010-4485/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2012.10.039

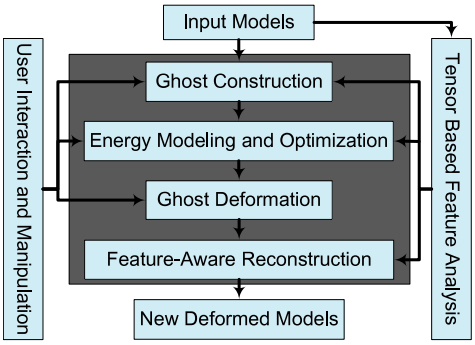

Fig. 1. The functional pipeline of our new method.

the convergence. Finally, an efficient feature-aware reconstruction is applied to map the deformation back to the original mesh. Our interface is simple: only the click-and-drag is needed. To the best of our knowledge, our interface enforces the simplest constraint during shape deformation while retaining feature-preserving and structure-preserving properties. Fig. 1 illustrates the pipeline of our approach. Note that the ghost is transparent from an ordinary user's perspective, and manipulation of the ghost can be abstracted as a black-box for general users.

2. Deformation modeling

In this section, we first describe the feature analysis method (prior to shape deformation), and then we formulate the deformation as a nonlinear optimization problem subject to constraints.

2.1. Tensor-based feature analysis

Given a 2D manifold M, let (V, E, F) be vertex set V, edge set E, and face set F that comprise an irregular triangular mesh of M. We introduce a feature analysis process based on local structure tensor.

Structure tensors are usually used to detect the local features of a mesh because of their informativeness [10,11]. Compared with a traditional feature analysis metric, such as a curvature or a normal, the tensor is more robust, and it contains both geometry information and direction information, which can help distinguish weak features from noise. Also, they can be easily extended to high-dimensional feature analysis. Specifically, a normal voting tensor $\mathbf{T}(v_i)$ of a vertex v_i can be expressed as

$$\mathbf{T}(v_i) = \sum_{t_j \in N_t(v_i)} \mu_j \mathbf{n}_{t_j} \mathbf{n}_{t_j}^T, \tag{1}$$

where t_j is a triangle, $N_t(v_i)$ denotes the set of neighboring triangles of v_i, $\mathbf{n}_{t_j}$ is the normal of t_j, and μ_j is the weight coefficient. Since we prefer a scaling- and stretching-invariant tensor, the weight works well with a constant value 1 in our approach. Note that the structure tensors of the original mesh are only computed once, and they can be reused in later processing.

According to the eigen-analysis of the structure tensor in Eq. (1) and the neighboring relationship, we extract and analyze features of the original mesh by generalizing the multi-type feature classification proposed in [10,11]. We treat the eigenvector corresponding to the smallest eigenvalue as the diffusion direction which is used to define "neighboring vertex coincidence" (NVC) in [11]. We could utilize these attributes to detect both prominent and weak features, and classify the vertices into different types of feature. For each vertex of the mesh, $\lambda_1 > \lambda_2 > \lambda_3 \geq 0$ are eigenvalues of the corresponding structure tensor, then the analysis is documented as

- Planar: if λ_1 is dominant, and λ_2 and λ_3 are close to 0.
- Corner: if λ_1, λ_2 and λ_3 are approximately equal.
- Sharp: if λ_1 and λ_2 are dominant, λ_3 is close to 0, and the NVC is satisfied.
- Weak: if λ_1 is dominant, λ_2 is weak, λ_3 is close to 0, and the NVC is satisfied.

The vertices of the mesh are then classified into different types, including planar vertices, corner vertices, sharp feature vertices, and weak feature vertices. The different types of vertex are treated differently in later processing, which will help our framework acquire the information of features and structures. To better adapt open meshes, boundary vertices are also treated as features, and the corresponding diffusion direction is along the boundary curve.

2.2. Energy formulation

Structure energies. The structure tensor formulated in Eq. (1) contains ample information; it can well depict the local geometry. Moreover, it has the direction information, which implies the structure characteristic. We exploit the structure tensor to preserve the features and structures of a model via a quadratic energy. The tensor is a positive semi-definite tensor of second order; consequently, it can be diagonalized by the eigenvalues $(\lambda_1 > \lambda_2 > \lambda_3 \geq 0)$. We reformulate it by a spectral representation

$$\begin{aligned}\mathbf{T}(v_i) = {} & \widetilde{\lambda}_1 \mathbf{e}_1 \mathbf{e}_1^T + \widetilde{\lambda}_2 (\mathbf{e}_1 \mathbf{e}_1^T + \mathbf{e}_2 \mathbf{e}_2^T) \\ & + \widetilde{\lambda}_3 (\mathbf{e}_1 \mathbf{e}_1^T + \mathbf{e}_2 \mathbf{e}_2^T + \mathbf{e}_3 \mathbf{e}_3^T),\end{aligned} \tag{2}$$

where $\mathbf{e}_i$ is the corresponding eigenvector of λ_i, $i = 1, 2, 3$, $\widetilde{\lambda}_1 = \lambda_1 - \lambda_2$, $\widetilde{\lambda}_2 = \lambda_2 - \lambda_3$, and $\widetilde{\lambda}_3 = \lambda_3$. Specifically, $\mathbf{e}_1 \mathbf{e}_1^T$ describes a stick, $\mathbf{e}_1 \mathbf{e}_1^T + \mathbf{e}_2 \mathbf{e}_2^T$ describes a plate, and $\mathbf{e}_1 \mathbf{e}_1^T + \mathbf{e}_2 \mathbf{e}_2^T + \mathbf{e}_3 \mathbf{e}_3^T$ describes a ball. Therefore, the local shapes can be well depicted by these structural attributes with proper parameters $\widetilde{\lambda}_i$, $i = 1, 2, 3$. These parameters can be viewed as a representation of local shapes; we call them structure parameters (SPs).

Given a discrete formulation of the SPs $\{\widetilde{\lambda}_i(v), i = 1, 2, 3\}$, we strive to comply with the prescribed structures in a least-squares sense, and the structure energy E_s is defined as

$$E_s = \sum_{v_i \in V} \sum_{j=1}^{3} \omega_i (\widetilde{\lambda}'_j(v_i) - \widetilde{\lambda}_j(v_i))^2, \tag{3}$$

where $\widetilde{\lambda}_j(v_i)$ is the j-th SP of vertex v_i, $\widetilde{\lambda}'_j(v_i)$ is the unknown SP, and ω_i is the weight, with default 1 for planar vertices, 2 for feature vertices (sharp and weak), and 3 for corner vertices. Although an SP alone does not uniquely define a surface, it is a powerful tool for preserving the structure when editing models. For some structure-stretching deformations, such as extending a cylinder while keeping the diameter unchanged, this is hard to achieve with very few point constraints. To accelerate the convergence speed and better maintain the structure, face normals can be considered as an optional assistant for this kind of deformation. The face normal energy is defined as

$$E_{fn} = \sum_{f_i \in F} (\mathbf{n}_{f'_i} - \mathbf{n}_{f_i})^2, \tag{4}$$

where f_i is the face of original mesh, and $\mathbf{n}_{f_i}$ is the corresponding normalized face normal. In other deformations, such as rotation and bending, the face normal energy should be assigned a small weight.

Metric energies. The metric terms E_{area} and E_{con} are used to quantify the deviation from original area and angle [2,3].

$$E_{area} = \sum_{f_i \in F} \frac{1}{A_{f_i}} (A_{f'_i} - A_{f_i})^2, \tag{5}$$

$$E_{con} = \sum_{f_i \in F} \sum_{j=1}^{3} \left(\frac{A_{f_i}}{3 \|\mathbf{e}_{f_i,j}\|^2} \frac{\|\mathbf{e}_{f'_i,j}\|^2}{A_{f'_i}} + \frac{\boldsymbol{\phi}_{f_i}^2}{12 A_{f_i}} \frac{\|\mathbf{h}_{f'_i,j}\|^2}{A_{f'_i}} \right), \tag{6}$$

where A_{f_i} is the face area, $\mathbf{e}_{f_i,j}$ is the edge adjacent to the face, $\boldsymbol{\phi}_{f_i}$ is the circumference of the face, and $\mathbf{h}_{f_i',j}$ is the vector perpendicular to the edge and pointing to the center of the inscribed circle of f_i'. They measure the areal and conformal distortion, respectively. As mentioned in [2], a weighted sum of the two energies provides more flexibility to control the deformation semantics. We use the metric energies to maintain good mesh quality during energy optimization.

Total energy. The total surface energy on a mesh is computed by a weighted sum of the above energies:

$$E_T = k_s E_s + k_{fn} E_{fn} + k_{area} E_{area} + k_{con} E_{con}, \tag{7}$$

where k_s, k_{fn}, k_{area}, and k_{con} are the scalar weights. A dominant structure weight leads to the strict structure preservation, while uniform scaling and parallel stretching are allowed. A strong face normal term yields great support to the structure preservation for uniform scaling and parallel stretching operations. A dominant area weight leads to a constrained deformation with fairly strong distortion of angles and curvatures, while a dominant conformal term leads to a uniform scaling. Users can adjust the scalar weights according to the specific application and the properties of the original structure; this enables versatile surface editing and shape design.

2.3. Deformation constraints

A preferred deformation can be obtained by combining the above surface energies and some necessary deformation constraints. Since our framework depicts the features and structures, we would like to achieve the desired results using more intuitive operations and as few constraints as possible. As a solution, a simple click-and-drag interface is utilized, which will be illustrated later. We formulate the constraints using the least-squares energy

$$E_c = \sum_{\widehat{v}_i \in C} \|v_i' - \widehat{v}_i\|^2, \tag{8}$$

where $\widehat{v}_i \in C$ is the point-handle constraint, and v_i' is the corresponding vertex of unknown mesh.

For local structure editing, specific shape constraints of the particular parts are often required rather than simple position constraints. This type of deformation can be achieved by simply specifying a few local reference vertices as shape constraints. For each special local structure, the local constraints can be formulated as

$$E_{lc} = \sum_{\widehat{v}_i \in C_l} \|(\widehat{v}_i' - c') - \alpha(\widehat{v}_i - c)\|^2, \tag{9}$$

where $\widehat{v}_i \in C_l$ is the constraint for the local structure of original mesh, c is the barycenter of constraints in C_l, α is the scaling rate with default value 1, and $\widehat{v}_i'$ and c' are the constraint and its corresponding barycenter of unknown mesh, respectively.

2.4. Optimization

The vertex positions of the deformed mesh are computed by minimizing the energies comprising the constraints and the energies introduced in the previous section, such that

$$\{v_1^*, \ldots, v_n^*\} = \underset{v_i', \ldots, v_n'}{\operatorname{argmin}} \sum_i k_i E_i, \tag{10}$$

where k_i and E_i are the weights and the corresponding different energies. This nonlinear least squares problem is solved by a Gauss–Newton solver [12], which enables superlinear convergence without the need for computing second-order derivatives.

3. Structure-preserving deformation

Solving the optimization problem directly on original meshes is expensive. It is challenging to achieve an interactive rate, even for models with only thousands of vertices. To reduce the complexity of this problem, we restrict the optimization to a ghost, which is a feature- and structure-aware subset of the original mesh. After the optimization, an efficient feature-aware reconstruction is utilized to obtain the deformed mesh in original space.

Algorithm 1: Ghost construction.

```
input  : Original mesh M, parameters.
output: Ghost G.
Initialize: G = M;
while at least one vertex can be removed or under the specified
iterations do
    set vertices of G unlabeled;
    for each vertex v of G do
        if v is labeled or a corner vertex then
            continue;
        else if v is a planar vertex then
            find connected edge with the least weight;
            if satisfy collapse condition then
                update G;
            end
        else if v is a sharp or weak feature vertex then
            if homologous is required then
                find the connected consistent feature edge
                with the least weight;
                if satisfy collapse condition then
                    update G;
                end
            end
        end
    end
end
```

3.1. Hierarchical ghost construction

A ghost is a feature subspace of a mesh, which is constructed to approximate the original mesh space, while preserving the essential structures. The ghost can be built using model simplification. For our specific problem, we require the ghost to be a subset of the original mesh, to have the same structure and topology, to be hierarchical for different types of deformation, and at the same time to have no long and skinny triangles (area close to 0). There are numerous methods for mesh simplification [13,14]; however, to the best of our knowledge, there is no method that is immediately available that satisfies all the above requirements. We tackle this challenge by exploring a hierarchical simplification algorithm driven by our feature analysis.

Our simplification is based on edge collapses, which selectively decimate the vertices in a hierarchical way. The main advantage of our method is that different types of vertex are treated differently according to the previous knowledge of features and structures. The hierarchical simplification includes homologous (feature vertices and non-feature vertices are harmoniously processed) and non-homologous (feature vertices and non-feature vertices are handled separately) simplifications. Users can choose the ghost to be homologous or non-homologous according to the type of model. For rigid manufactured models, stretching or scaling are the most common operations, and a non-homologous ghost is adequate to meet such conditions. For more general models, the ghost must have adequate constituents to express the particulars when rotation occurs; hence a homologous ghost is needed. More details are documented in Algorithm 1. Some notation should be explained here. The edge weight is set to its length in our experiments. The collapse condition includes the following. (1) For

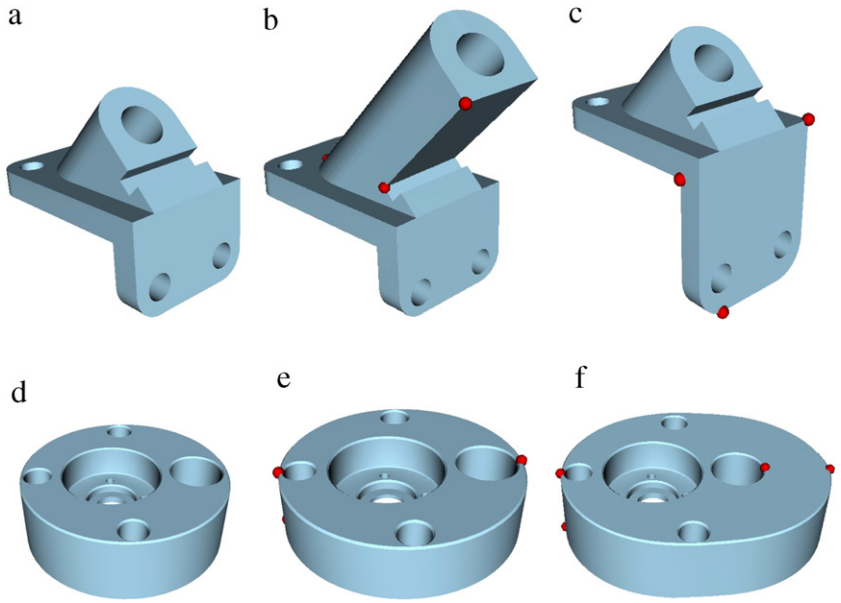

Fig. 2. Structure-preserving deformation for mechanical models (anchor and coupling). (a)–(c) The stretching deformations. (d)–(f) The multi-scale structure-preserving deformations.

each related face of the detected edge, the new area is larger than a threshold (we set it as 0.1*(the average area of all the related faces)). (2) For each related face, the dot product of the new normal and the old one is larger than a threshold (we set it as 0.98 here). The collapse condition guarantees the quality and the geometric accuracy of ghost. The edge is called a consistent feature edge if and only if the neighboring vertex is a non-planar vertex, and, at the same time, the NVC condition is satisfied. The update process includes collapsing the detected edge, deleting the current vertex, labeling the neighboring vertices, and adjusting the connectivity.

3.2. Feature-aware reconstruction

After the deformation optimization (expressed in Eq. (10)) is solved on the ghost, a reconstruction is necessary to map the deformation back to the original mesh. The reconstruction must be feature aware in order to better preserve the features and structures, and at the same time the efficiency has to be considered. In this section, we will develop a linear reconstruction system to combat this problem effectively.

Taking all factors into consideration, we choose the normal-controlled coordinates (NCCs) proposed in [15] to achieve the reconstruction for several reasons, including they are always parallel to the corresponding normals, they well encode local geometric details, and they can express the reconstruction as a linear problem. From the construction of the ghost, we know it is a subset of the original mesh, which means that each vertex on the ghost corresponds to a vertex on the original mesh. Therefore it is technically feasible to estimate vertex normals of the deformed mesh from the deformed ghost. Then the target NCCs can be updated with the new normals according to the parallel property. Hence, the deformed mesh can be constructed via linear reconstruction

$$\begin{bmatrix}\mathbf{N}\\ \mathbf{I}\end{bmatrix}\mathbf{V}' = \begin{bmatrix}\boldsymbol{\delta}\\ \mathbf{S}\end{bmatrix}, \tag{11}$$

where $\mathbf{V}'$ is the matrix of unknown vertices, $\mathbf{N}$ is a sparse matrix constructed by the NCC parameters, $\boldsymbol{\delta}$ consists of the updated NCCs, and $\mathbf{S}$ is the corresponding constraint matrix that comprises the vertices of ghost. To better satisfy the feature-aware requirement, we further modify the NCCs of prominent feature vertices. When a vertex is a feature, only its non-planar neighboring vertices are involved in the computation. If there are fewer than three neighbors, we simply record the NCCs as the differences between the current vertex and the centroids of its neighboring vertices.

4. Experimental results and discussion

We now demonstrate the performance of our method by conducting experiments in various aspects. All the experiments documented in this paper were conducted on a computer with 1.6 GHz Intel Core (TM, 4Core/8Threads) i7 CPU with 4G RAM. Most computation expenses of our approach, such as the tensor computation, feature analysis, ghost construction, and Cholesky decomposition of the reconstruction matrix, can be done in the preprocessing stage, and only the back-substitution procedure for recomputation is needed. The ghost is invisible from the user's perspective, but the corresponding vertices on the original mesh are highlighted as candidates for direct control, and the user only needs to operate on the original mesh.

Structure-preserving deformation. Fig. 2(a)–(c) illustrate the structure-preserving deformation for mechanical models. For this kind of model, stretching and scaling are the most common operations. The user simply selects a certain number of candidates as handles (red balls) that can help drive or maintain the structure-preserving deformation. Fig. 2(d)–(e) show a greedy multi-scale structure-preserving deformation. In this example, the uniform stretching and scaling occurs when the user drags the handles along the horizontal direction (e). Also, the user controls deformation scaling of the particular structure by simply adding some control handles at the reference positions (f).

Our deformation framework can also handle general scanned models. For this type of model, features are not clearly discrimitive, and deformations usually contain scaling, stretching, and rotation. Therefore, appropriately increasing constraint handles can settle this issue. For example, the necessary constraints for key positions of deformation should be selected if the deformation is complex. Fig. 3 shows the structure-preserving deformation for general models.

Comparison with related work. We compare our method with two excellent related deformation schemes proposed in [2,3]. Our method can be viewed as an extended improvement of the

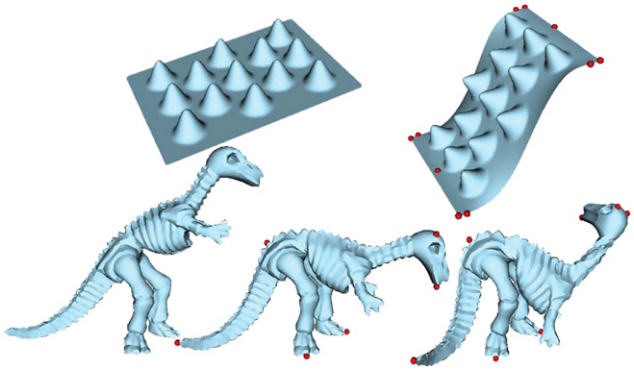

Fig. 3. General structure-preserving deformations. The original bumpy plane and dinosaur models (left). A couple of constraints are selected to control the complex shape deformation (right).

algorithm in [2], and we focus more on the structure-preserving aspect, the ease of operation, and the improved efficiency. Our method can significantly improve the computational efficiency by transferring the optimization problem into a feature subspace. The method in [3] also uses a subspace to improve the efficiency; however, their calculations are more complex. Moreover, since there is no prior knowledge of the structures, they need more region constraints to control their energy optimization. Fig. 4 shows the comparisons between our method and the methods in [2,3]. Table 1 lists the time statistics in our approach: (from left to right) vertex number of the original meshes (Data), vertex number of the ghosts (Ghost), the time (in seconds) for one iteration (Solve), the reconstruction (Recon.), and the preprocess (Prep.).

Table 1
Statistics and performance time (s) of selected models.

Data (# V)	Ghost	Solve	Recon.	Prep.
Fandisk (6k)	0.1k	0.001	0.024	36
Coupling (12k)	1k	0.009	0.059	78
Bumpy plane (12k)	1.5k	0.014	0.061	82
Dinosaur (32k)	2k	0.018	0.207	216
Vase (100k)	1k	0.009	0.648	771

Local structure-preserving deformation. Local structure editing is also supported in our framework. For this type of deformation, the user only needs to pick a few constraints around the special structures, then use the local structure constraints defined in Eq. (9). Fig. 5 shows the car models in which the size of the front window (top) and the wheels (bottom) are well kept when editing the global shapes. We also support more versatile editing by directly prescribing the scaling rate of local structures.

5. Conclusion

In this paper, we have articulated a novel method for structure-preserving deformation with a simple user interface. The central idea is the unification of shape analysis and deformation towards acquiring high-level knowledge about the shape to be edited. The feature and structure information is fully utilized for ghost

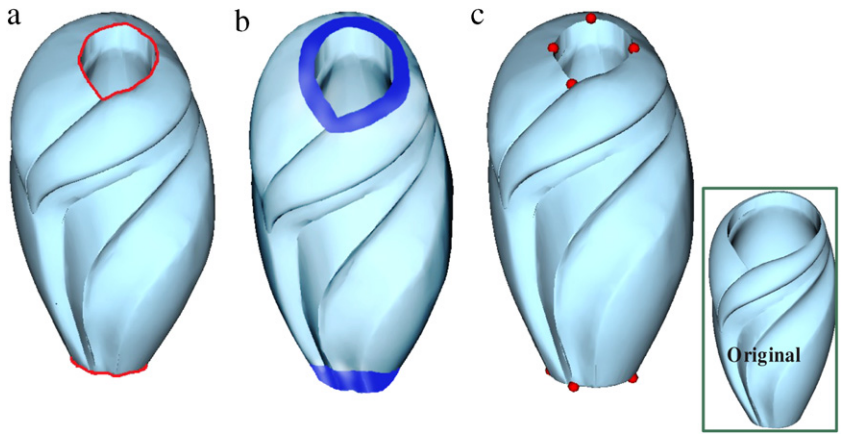

Fig. 4. A comparison of the results produced by related methods. (a) The deformation in [2], whose constraints are feature lines colored in red. (b) The deformation in [3], whose constraints are specified regions colored in blue. (c) Our method, which produces a comparable result to (a) and (b), while using the simplest and fewest constraints highlighted by the red balls. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

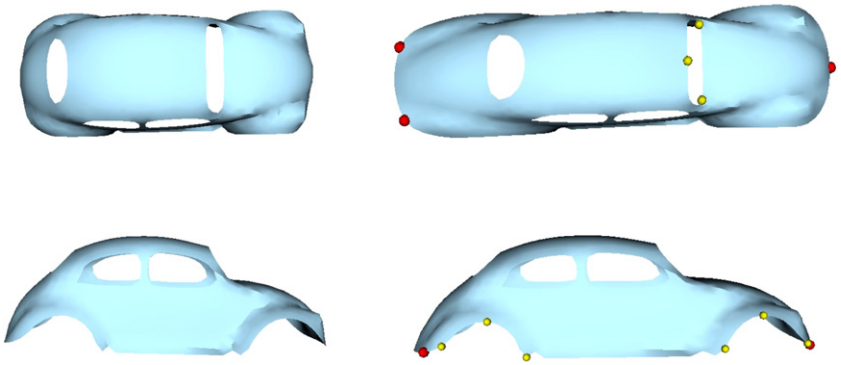

Fig. 5. Local structure-preserving deformation. Left: the original car models. Right: the deformed car models with front window and wheels unchanged. The yellow balls are local structure (shape) constraints, and the red balls are the handle (position) constraints. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

construction, surface energy modeling, and deformation reconstruction. Our deformation framework also supports local structure-preserving editing with a greedy constraint. It enhances existing approaches for structure manipulation and leads to versatile surface deformations for shape design.

For immediate future work, we are planning to extend our method to handle diverse types of geometric, material, and scientific data. Moreover, automatic definition of potential constraint points and new structural design deserve further investigation, since they could significantly broaden the range of application of our method.

Acknowledgments

This research is supported in part by the Fundamental Research Fund for the Central Universities, National Natural Science Foundation of China–Guangdong Joint Fund grant U0935004, National Natural Science Foundation of China grant 60873181, and US NSF grants IIS-0710819, IIS-0949467, IIS-1047715, and IIS-1049448.

References

[1] Yu Y, Zhou K, Xu D, Shi X, Bao H, Guo B, Shum H-Y. Mesh editing with Poisson-based gradient field manipulation. ACM Transactions on Graphics 2004;23:644–51.
[2] Eigensatz M, Pauly M. Positional, metric, and curvature control for constraint-based surface deformation. Computer Graphics Forum 2009;28(2):551–8.
[3] Hildebrandt K, Schulz C, Tycowicz CV, Polthier K. Interactive surface modeling using modal analysis. ACM Transactions on Graphics 2011;30(5):1–11.
[4] Miropolsky A, Fischer A. Utilizing diverse feature data for reconstruction of scanned object as a basis for inspection. ACM Transactions on Graphics 2007;7:211–24.
[5] Kolomenkin M, Shimshoni I, Tal A. On edge detection on surfaces. In: Computer vision and pattern recognition. 2009. p. 2767–74.
[6] Gal R, Sorkine O, Mitra NJ, Cohen-or D. iwires: An analyze-and-edit approach to shape manipulation. ACM Transactions on Graphics 2009;28(2):1–10.
[7] Sederberg TW, Parry SR. Free-form deformation of solid geometric models. In: SIGGRAPH. 1986. p. 151–60.
[8] Yoshizawa S, Belyaev AG, Seidel H-P. Free-form skeleton-driven mesh deformations. In: ACM symposium on solid modeling and applications. 2003. p. 247–53.
[9] Ben-Chen M, Weber O, Gotsman C. Variational harmonic maps for space deformation. ACM Transactions on Graphics 2009;28:1–11.
[10] Kim HS, Choi HK, Lee KH. Feature detection of triangular meshes based on tensor voting theory. Computer-Aided Design 2009;41(1):47–58.
[11] Wang S, Hou T, Su Z, Qin H. Diffusion tensor weighted harmonic fields for feature classification. In: Pacific graphics. 2011. p. 93–8.
[12] Madsen K, Nielsen H, Tingleff O. Methods for non-linear least squares problems. Technical report, Technical University of Denmark. 2004.
[13] Hoppe H. Progressive meshes. In: Computer graphics and interactive techniques. 1996. p. 99–108.
[14] Luebke DP. A developer's survey of polygonal simplification algorithms. IEEE Computer Graphics and Applications 2001;21:24–35.
[15] Wang S, Hou T, Su Z, Qin H. Multi-scale anisotropic heat diffusion based on normal-driven shape representation. The Visual Computer 2011;27(6–8):429–39.