A Unified Shape Editing Framework Based on Tetrahedral Control Mesh

Abstract

It is a fundamental but challenging problem to efficiently edit complex 3D objects. By embedding the input models into coarse tetrahedral control meshes, this paper develops a unified framework to discuss two useful editing operations: interactive deformation and deformation transfer. First, a new rigidity energy is proposed to make the tetrahedral control mesh deform as rigidly as possible, which yields intuitive detail and volume preservation even under large deformations. And an error-driven refinement approach is presented to further improve the deformation result. Then, based on this deformation scheme, a volumetric correspondence method is introduced to perform deformation transfer task between the tetrahedral control meshes of the source and target models, which greatly lessens the burden of the user. Experimental results show our algorithm is effective, easy to control, supports various shape representations, and well transfers deformations between non-homeomorphous models.

Keywords: deformation, deformation transfer, tetrahedral control mesh, rigidity

Introduction

In recent years, complex 3D models are widely used in many application fields, e.g., industry design, digital entertainment, and military simulation. One of the most challenging geometry processing operations is high quality shape editing. In this paper, we mainly discuss two kinds of editing operations: interactive deformation and deformation transfer.

Interactive deformation manipulates a given model under the guidance of the user. Deformation transfer clones deformations exhibited by a source model to a different target model. The challenging problems are preserving geometric details and preventing unnatural volumetric distortions as much as possible while satisfying the user constraints.

There are a huge number of techniques developed for interactive deformation and deformation transfer. However, most of them are surface-based, because 3D models are conventionally represented with surface meshes. The surface-based techniques only consider surface detail preservation. So certain volumetric features, such as local rigidity and volume, are difficult to preserve, and apparent volume changes occur during extreme shape editing. In addition, directly processing high resolution surface meshes suffers from low performance. This paper develops a simple yet powerful framework to address these issues.

Once the model to be deformed is embedded into a coarse tetrahedral control mesh, we present a new rigidity energy to deform the control mesh as rigidly as possible, then pass the deformation to the embedded model by the modified barycentric interpolation [1].

Enforcing rigidity of the control mesh leads to a stable numerical solver, and prevents the embedded model from detail distortions and volume changes even under large deformations,

as illustrated in Figure 1. However, the original control mesh may be too coarse to fine-scale deformations, we propose an error-driven refinement scheme to further optimize the high deformation error regions.

As an extension, a novel deformation transfer algorithm is presented: the transfer process is performed between the source and target tetrahedral control meshes instead of between the source and target models. To achieve this goal, a volumetric correspondence method is introduced to guide the transfer process, and a fitting process is proposed to obtain the deformation of the source control mesh.

Compared with existing methods, our transfer algorithm requires only a few marker pairs to build the correspondence, and greatly lessens the burden of the user.

Using coarse tetrahedral control mesh to drive the editing of complex 3D objects makes our framework achieves interactive performance, accommodates a variety of shape representations, such as meshes, point clouds, models with multiple parts or non-manifold structures etc, and easily transfers deformations between non-homeomorphous models.

Related work

There has been considerable amount of research work for interactive deformation and deformation transfer.

Freeform deformation (FFD) techniques [1, 2, 3, 4] embed the object into a simpler domain, then the user can deform the object by freely editing the domain. Although FFD is

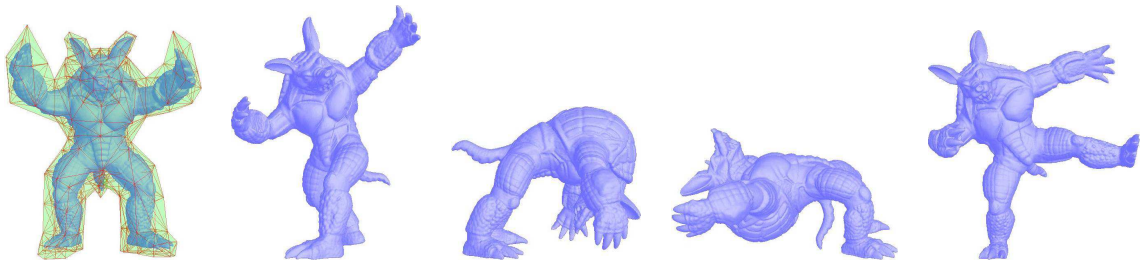


Figure 1: Deformation results of Armadillo. The leftmost are the original Armadillo and its tetrahedral control mesh, the others are results with different poses. The rigidity energy effectively preserves geometric details and prevents unnatural volumetric distortions.

independent of object representations, it is hard to maintain details.

Multi-resolution techniques [5, 6, 7] decompose a given mesh into a sequence of representations with decreasing level of detail. Deformation is obtained by alternating the base model and transferring back the details. The problem is that details are not coupled and preserved uniformly over the whole model, so artifacts still appear in highly deformed regions.

Gradient domain deformation techniques [8, 9, 10, 11, 12, 13, 14] cast mesh deformation as an optimization problem, and reconstruct the deformed mesh by fitting the alternated differential coordinates representing local details. However, these algorithms require solving large linear systems, thus time and memory cost are very expensive.

Recently, a new deformation strategy is proposed: the 3D shape is constrained to undergo an as-rigid-as-possible deformation by asking for rigidity of its local behavior. Our method also falls into this category. Botsch et al. [15] relate the mesh in a layer of elastically coupled prisms that envelop it. During deformation, the prisms are rigidly transformed to

satisfy user constraints and prevent the mesh from degenerations. As the number of prisms is comparable to the face number of the mesh, its bottleneck is the computational performance. Then they [16] extend this idea to adaptive volumetric hexahedral cells: the volumetric cells are treated as rigid objects, and the integrated distance between their adjacent faces is minimized. Although it achieves generality, the limitation is also its performance because of solving dense linear systems. Sorkine et al. [17] rigidly deform each edge of the mesh to avoid local distortions. Nevertheless, directly deforming detailed meshes may suffer from slow convergence. Sumner et al. [18] induce the deformation by a collection of rigid transformations organized in a graph structure. However, its spatial nature may result in part of the graph influencing an undesired area of the shape. River et al. [19] propose fast lattice shape matching with user-specified stiffness for realtime simulation. But the uniform space discretization scheme couldn't reflect the irregularity of the embedded geometry.

Deformation transfer is first introduced by sumner et al. [20], aiming at deformation reuse. It encodes the deformation as a set of simplex transformations, which are used for letting a static target mesh follow the deformation sequence of a source mesh. Zayer et al. [21] perform the transfer task with the help of harmonic field and Possion editing. Chang et al. [22] clones a skeleton-driven animation to another character model. Wang et al. [23] provide an easy-to-use animation system for non-professionals. Since these methods all directly perform the transfer task between the source and target models, a surface correspondence is required to guide how source deformations are transferred. Note that the quality of correspondence strongly depends on user-specified maker pairs. But for densely

sampled models, it's a very tedious and time-consuming task to provide enough maker pairs.

Interactive Deformation

We first present our deformation technique. Let M be the model to be deformed, and $C = (U, H)$ be its coarse tetrahedral control mesh, where $U = (\mathbf{u}_1, ..., \mathbf{u}_n)$ denotes the set of n control vertices and $H = (h_1, ..., h_m)$ denotes the set of m tetrahedra.

Modified Barycentric Interpolation

We embed M into C by using a modified barycentric interpolation (modified-BI) [1] to establish their relationship.

To eliminate the first-order discontinuity artifacts of traditional barycentric interpolation, the modified-BI adjusts the deformation gradients to be as close as possible between adjacent tetrahedra by introducing a linear transformation for each control vertex.

For a point $\mathbf{u}$ in the interior of C, the modified-BI is formulated as:

$$\mathbf{x}(\mathbf{u}) = \sum_{i=1}^{n} \phi_i(\mathbf{u})(\mathbf{x}_i + \mathbf{M}_i(\mathbf{u} - \mathbf{u}_i)), \tag{1}$$

where $\phi_i(\mathbf{u})$ is the barycentric coordinate basis function, and $\mathbf{x}_i$, $\mathbf{M}_i$ are respectively the deformed position and linear transformation of control vertex $\mathbf{u}_i$.

We set all deformed vertex positions $\{\mathbf{x}_i\}_{i=1}^n$ and linear transformations $\{\mathbf{M}_i\}_{i=1}^n$ as vector $\mathbf{x}$ and vector $\mathbf{m}$ respectively. $\{\mathbf{M}_i\}_{i=1}^n$ are obtained by optimizing a sum of two quadratic linear energies: discontinuity energy $E_{disc}(\mathbf{x}, \mathbf{m})$ and vibration energy $E_{vibr}(\mathbf{x}, \mathbf{m})$.

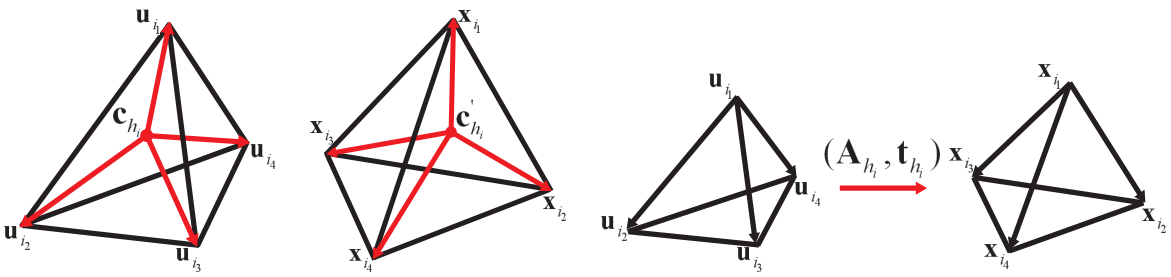


Figure 2: Demonstration of rigidity energy. (a) original tetrahedron, (b) deformed tetrahedron, (c) deformation estimation.

Rigidity Energy

However, as the modified-BI is not conformal, M suffers from detail distortions and volume changes during deformation. To prevent these artifacts, a new rigidity energy is proposed to deform C as rigidly as possible.

As shown in Figure 2, when a tetrahedron $h_i = (\mathbf{u}_{i_1}, \mathbf{u}_{i_2}, \mathbf{u}_{i_3}, \mathbf{u}_{i_4})$ undergoes a rigid deformation, there exists a rotation matrix $\mathbf{R}_{h_i}$ such that

$$\mathbf{x}_{i_j} - \mathbf{c}'_{h_i} = \mathbf{R}_{h_i}(\mathbf{u}_{i_j} - \mathbf{c}_{h_i}), \qquad 1 \leq j \leq 4$$

where $\mathbf{x}_{i_j}$ is the deformed position of $\mathbf{u}_{i_j}$, and $\mathbf{c}_{h_i}$, $\mathbf{c}'_{h_i}$ are the centroids of original and deformed tetrahedra.

Since the deformation actually is not rigid, the above equations can only be satisfied in the least squares sense, i.e., minimizing

$$E_{h_i}(\mathbf{x}) = \sum_{j=1}^{4} \|(\mathbf{x}_{i_j} - \mathbf{c}'_{h_i}) - \mathbf{R}_{h_i}(\mathbf{u}_{i_j} - \mathbf{c}_{h_i})\|^2. \tag{2}$$

Obviously, the smaller $E_{h_i}(\mathbf{x})$, the more rigidly h_i deforms.

Summing $E_{h_i}(\mathbf{x})$ for all tetrahedra in H, we get the rigidity energy of C:

$$E_{rigidity}(\mathbf{x}) = \sum_{h_i \in H} w_i E_{h_i}(\mathbf{x}), \tag{3}$$

where w_i is the weight for h_i, and we simply take its volume $|h_i|$ into account.

Now we briefly describe the derivation for these optimal rotation matrices $\{\mathbf{R}_{h_i}\}_{i=1}^{m}$. Because h_i determines $\{\mathbf{u}_{i_j} - \mathbf{c}_{h_i}\}_{j=1}^{4}$, the rotation part of its deformation $(\mathbf{A}_{h_i}, \mathbf{t}_{h_i})$ can be used for approximating $\mathbf{R}_{h_i}$ (Figure 2(c)), where $\mathbf{A}_{h_i}$ is a linear transformation encoding the change in orientation and scale, $\mathbf{t}_{h_i}$ is a translation vector encoding the change in position.

According to Müller et al. [24], the linear transformation $\mathbf{A}_{h_i}$ should satisfy $\mathbf{A}_{h_i}\mathbf{U}_{h_i} = \mathbf{X}_{h_i}$, where $\mathbf{U}_{h_i} = (\mathbf{u}_{i_2} - \mathbf{u}_{i_1}, \mathbf{u}_{i_3} - \mathbf{u}_{i_1}, \mathbf{u}_{i_4} - \mathbf{u}_{i_1})$ and $\mathbf{X}_{h_i} = (\mathbf{x}_{i_2} - \mathbf{x}_{i_1}, \mathbf{x}_{i_3} - \mathbf{x}_{i_1}, \mathbf{x}_{i_4} - \mathbf{x}_{i_1})$.

As $\mathbf{U}_{h_i}$ is a nonsingular matrix, thus

$$\begin{cases} \mathbf{A}_{h_i} = \mathbf{X}_{h_i}\mathbf{U}_{h_i}^{-1} \\ \mathbf{t}_{h_i} = \mathbf{c}_{h_i}^{'} - \mathbf{A}_{h_i}\mathbf{c}_{h_i} \end{cases}. \tag{4}$$

And the rotation part can be obtained by performing polar decomposition to $\mathbf{A}_{h_i}$:

$$\mathbf{R}_{h_i} = \mathbf{A}_{h_i}\sqrt{\mathbf{A}_{h_i}^{T}\mathbf{A}_{h_i}}^{-1}. \tag{5}$$

Position Constraint

To guide the deformation, the user interactively selects some control vertices and directly moves them to the desired positions.

Let $\{c_i\}_{i=1}^k$ be indices of the manipulated control vertices, the position constraint is given as $E_{pos}(\mathbf{x}) = \sum_{i=1}^{k} \|\mathbf{x}_{c_i} - \widehat{\mathbf{x}}_{c_i}\|^2$, where $\widehat{\mathbf{x}}_{c_i}$ is the user-specified position for $\mathbf{u}_{c_i}$.

Thus the final optimization problem can be formulated as follows:

$$\min_{\{\mathbf{x},\mathbf{m}\}} \{E_{disc}(\mathbf{x},\mathbf{m}) + \alpha E_{vibr}(\mathbf{x},\mathbf{m}) + \beta E_{rigidity}(\mathbf{x}) + \gamma E_{pos}(\mathbf{x})\}, \tag{6}$$

where α, β and γ are weights balancing the four energy terms. These weights can be automatically determined such that the four energy terms are comparable, i.e., their Hessian matrices have the same norm.

Numerical Solver

The rigidity energy is a nonlinear constraint: on one hand, the discretization of $E_{rigidity}(\mathbf{x})$ requires the appropriately evaluated rotation matrices; on the other hand, the rotation estimation depends on the unknown deformed control mesh. So minimizing (6) is a chicken-and-egg problem. We adopt a two-phase method [25] to solve this problem.

Phase 1. overlook temporarily the rigidity energy, and optimize the remaining three linear constraints, i.e., the discontinuity energy, the vibration energy, and the position constraint. It equals solving a sparse linear system $\mathbf{P}\begin{pmatrix}\mathbf{x}\\ \\ \mathbf{m}\end{pmatrix} = \mathbf{b}$, where matrix $\mathbf{P}$, vector $\mathbf{b}$ are derived from $E_{disc}(\mathbf{x},\mathbf{m})$, $E_{vibr}(\mathbf{x},\mathbf{m})$ and $E_{pos}(\mathbf{x})$, and $\mathbf{P}$ relates only to the original control mesh.

Phase 2. minimize (6) based on the deformation result obtained from Phase 1 by iteratively performing the following two steps:

Step 1. Update the rotation matrices. Comparing the original and current state of the control mesh, a rotation matrix is computed for each tetrahedron.

Step 2. Update the vertex positions and linear transformations. Once $E_{rigidity}(\mathbf{x})$ is updated, a new sparse linear system $\begin{pmatrix} \mathbf{P} \\ \mathbf{Q} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{m} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \overline{\mathbf{b}} \end{pmatrix}$ is solved with respect to the current vertex positions and linear transformations, where matrix $\mathbf{Q}$, vector $\overline{\mathbf{b}}$ are derived from $E_{rigidity}(\mathbf{x})$, and $\mathbf{Q}$ relates only to the original control mesh.

After the iteration converges, the derived deformation is passed to the embedded model through the modified-BI. As the control mesh retains rigidity as much as possible during deformation, it prevents the embedded model from detail distortions and volume changes.

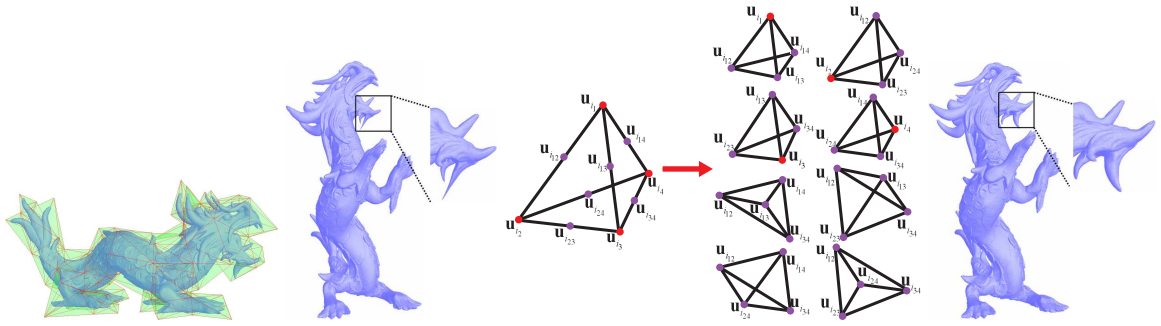
Error-driven Refinement

The original control mesh may be too coarse for fine deformations, as shown in Figure 3(b). Large deformation error at some regions necessitates subdividing the control mesh adaptively.

As the control mesh is required to behave as rigidly as possible, we define the deformation error of tetrahedron h_i by measuring its deviation from rigidity, i.e., the deviation of the current linear transformation $\mathbf{A}_{h_i}$ from its optimal rotation part $\mathbf{R}_{h_i}$:

$$E_{h_i}^{error} = \|\mathbf{A}_{h_i} - \mathbf{R}_{h_i}\|. \tag{7}$$

Whenever $E_{h_i}^{error}$ exceeds a user-specified threshold, h_i is subdivided to eight sub-tetrahedra



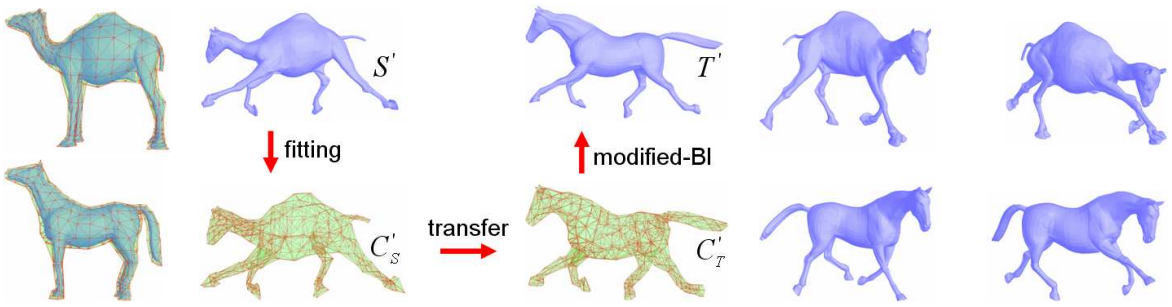
(a) (b) (c) (d)

Figure 3: Error-driven refinement. (a) original Asian dragon and its tetrahedral control mesh, (b) deformation result without refinement, note that its lower lip distorts, (c) refinement scheme: by inserting new vertices (in purple) at the midpoints of each edge and connecting them together, original tetrahedron $h_i = (\mathbf{u}_{i_1}, \mathbf{u}_{i_2}, \mathbf{u}_{i_3}, \mathbf{u}_{i_4})$ (in red) is subdivided to eight sub-tetrahedra: $(\mathbf{u}_{i_1}, \mathbf{u}_{i_{12}}, \mathbf{u}_{i_{13}}, \mathbf{u}_{i_{14}})$, $(\mathbf{u}_{i_{12}}, \mathbf{u}_{i_2}, \mathbf{u}_{i_{23}}, \mathbf{u}_{i_{24}})$, $(\mathbf{u}_{i_{13}}, \mathbf{u}_{i_{23}}, \mathbf{u}_{i_3}, \mathbf{u}_{i_{34}})$, $(\mathbf{u}_{i_{14}}, \mathbf{u}_{i_{24}}, \mathbf{u}_{i_{34}}, \mathbf{u}_{i_4})$, $(\mathbf{u}_{i_{12}}, \mathbf{u}_{i_{13}}, \mathbf{u}_{i_{14}}, \mathbf{u}_{i_{34}})$, $(\mathbf{u}_{i_{12}}, \mathbf{u}_{i_{13}}, \mathbf{u}_{i_{23}}, \mathbf{u}_{i_{34}})$, $(\mathbf{u}_{i_{12}}, \mathbf{u}_{i_{14}}, \mathbf{u}_{i_{24}}, \mathbf{u}_{i_{34}})$, $(\mathbf{u}_{i_{12}}, \mathbf{u}_{i_{23}}, \mathbf{u}_{i_{24}}, \mathbf{u}_{i_{34}})$, (d) deformation result with refinement.

to introduce more degrees of freedom for further optimization (Figure 3(c)). Figure 3(d) is the optimized result, we use 0.6 as the threshold. our refinement scheme automatically detects highly deformed regions and improves the result.

Deformation Transfer

As an extension, this section proposes a deformation transfer algorithm. In the following, let S, T be the source and target models, and C_S, C_T be their tetrahedral control meshes.



(a) (b) (c) (d)

Figure 4: Deformation transfer (Camel to Horse). (a) original models (S,T) and their tetrahedral control meshes (C_S,C_T), (b) transfer pipeline: $S^{'}$ is a deformed shape of S, the corresponding deformation of C_S is obtained through a fitting process; Then with tetrahedron correspondences and deformation scheme (6), the deformation is transferred to C_T and further passed to T by interpolation. (c)-(d) more transferred results.

Assuming $S^{'}$ is a deformed shape of S, the concept of deformation transfer can be understood as an analogy, i.e., generate a new model $T^{'}$ with respect to T such that the deformation exhibited by $T^{'}$ is analogous to that of $S^{'}$.

Unlike previous methods, we transfer shape deformation via that of their control meshes which are composed of sparse vertices. The transfer pipeline is shown in Figure 4(b). To attain this goal, a volumetric correspondence method is required to build the correspondence between C_S and C_T, and a fitting process is proposed to find a corresponding deformation of C_S that yields $S^{'}$.

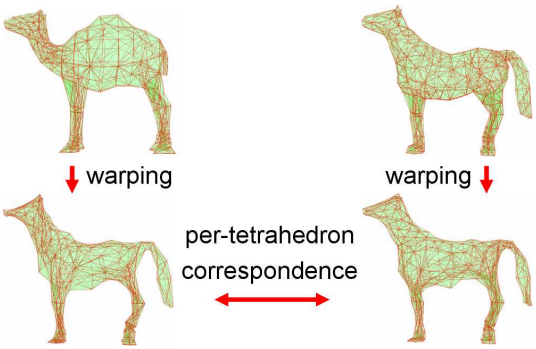


Figure 5: Volumetric correspondence. Tetrahedral control meshes C_S and C_T are first warped into similar shapes. Then the per-tetrahedron correspondence is determined by comparing the centroids of the warped tetrahedra.

Volumetric Correspondence

To mimic the deformation of the source object, we should build the correspondence between two control meshes C_S and C_T.

Wang et al. [23] proposes a surface correspondence method: the source and target meshes are first warped into similar shapes using least squares meshes [26], then triangle correspondences are determined by computing whether their centroids are close enough.

Now we extend this idea to tetrahedral control meshes (Figure 5). To aid the generation of the correspondence, some compatible markers should be specified on both control meshes. As C_S and C_T are very coarse, only a small number of marker pairs are needed, so it is easier to accomplish by the user.

In [23], least squares meshes actually solve a Laplace equation to reconstruct the warped results. In the volumetric case, the Laplace equation with marker constraints can be ex-

pressed as follows:

$$\begin{cases} (Div \cdot \Delta)\mathbf{x} = \mathbf{0} \\ \mathbf{x}_{m_i} = \widehat{\mathbf{x}}_{m_i} \quad 1 \leq i \leq l \end{cases}, \tag{8}$$

where Div, Δ are the divergence and gradient operators for tetrahedral mesh [27], $\{\widehat{\mathbf{x}}_{m_i}\}_{i=1}^{l}$ are marker constraints, $\{m_i\}_{i=1}^{l}$ are indices of user-specified markers, and l is the number of markers.

The above equation yields a harmonic map: it positions each control vertex approximately at the centroid of its immediate neighbors and makes the warped results fairly smooth except for regions near markers.

After C_S and C_T are warped into similar shapes, we determine the tetrahedron correspondences by comparing the centroids of the warped tetrahedra: a source tetrahedron and a target one are compatible if their centroids are close enough.

Fitting The Deformed Control Mesh

Suppose the control meshes of the deformed shape $S^{'}$, $T^{'}$ are $C_S^{'}$, $C_T^{'}$ respectively. During the deformation from S to $S^{'}$, its control mesh also undergoes a transform from C_S to $C_S^{'}$, which is the transform we want to share between C_T to $C_T^{'}$. The key problem now is to find $C_S^{'}$ based on the deformed shape $S^{'}$.

Note that the relationship between S and C_S is described by the modified-BI, and can be represented as $L(\mathbf{x}, \mathbf{m}, \mathbf{u})$, for $\mathbf{u} \in S$. A fitting energy is proposed to achieve the above goal by retaining their relationship during deformation:

$$E_{fitting}(\mathbf{x}, \mathbf{m}) = \oint_S \|S^{'}(\mathbf{u}) - L(\mathbf{x}, \mathbf{m}, \mathbf{u})\|^2 d\sigma, \tag{9}$$

where $S^{'}(\mathbf{u})$ is the deformed position of $\mathbf{u}$, and $d\sigma$ is the infinitesimal surface element around $\mathbf{u}$.

However, for control vertices of C_S whose neighboring tetrahedra don't intersect with S, they are not constrained by $E_{fitting}(\mathbf{x}, \mathbf{m})$, e.g., $E_{fitting}(\mathbf{x}, \mathbf{m})$ is under-constrained. Accounting for the rigidity constraint of each tetrahedron element during deformation, we finally optimize

$$\min_{\{\mathbf{x},\mathbf{m}\}} \{E_{fitting}(\mathbf{x}, \mathbf{m}) + \lambda E_{rigidity}(\mathbf{x})\} \tag{10}$$

to obtain the deformed control mesh $C_S^{'}$, where λ is a weight balancing the two energies.

The numerical solver is similar to that of optimization problem (6), except that the initial deformation result is generated using the linear constraint $E_{fitting}(\mathbf{x}, \mathbf{m})$.

Transfer Optimization

With the aid of volumetric correspondence and fitting process, the deformation from S to $S^{'}$ can be transferred to that from T to $T^{'}$.

In fact, by comparing C_S with $C_S^{'}$, we can represent the deformation as a collection of affine transformations $\{(\mathbf{A}_{h_i}^{C_S}, \mathbf{t}_{h_i}^{C_S})\}_{i=1}^{|C_S|}$ (formula (4)), where $\mathbf{A}_{h_i}^{C_S}$ is a linear transformation, $\mathbf{t}_{h_i}^{C_S}$ is a translation vector, and $|C_S|$ is the number of tetrahedra in C_S.

Because neighboring tetrahedra share vertices, directly copying these transformations to C_T will makes them seperate from each other [20]. To ensure that shared vertices be

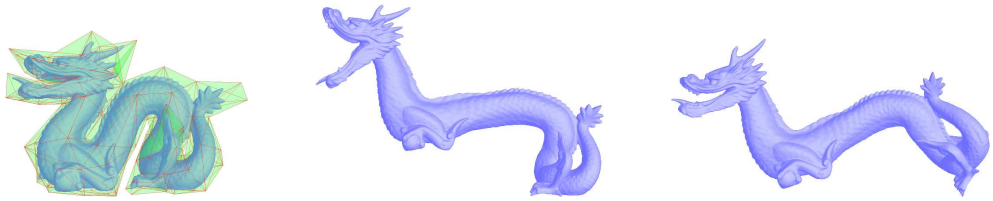


Figure 6: Deformation results of Dragon. The leftmost shows the original Dragon and its tetrahedral control mesh, the others are deformation results.

transformed to the same place, we apply deformation scheme (6) to maintain consistency taking directly transferred result as the initial value and user-specified markers as position constraint, and obtain the deformed target control mesh C_T'. Then the deformation is passed to T through the modified-BI.

Experimental Results and Discussions

We generate the coarse tetrahedral control meshes for the input models using an automatic mesh generator - NETGEN (http://www.hpfem.jku.at/netgen/). We implement our algorithm using C++ on a PC with 1.9GHz Pentium 4 CPU and 512MB memory.

Interactive Deformation Figure 1 and 6 show the effectiveness of the rigidity energy on complex models, where detail distortions and volume changes are well prevented.

Deformations on Santa with multiple parts and non-manifold structures are given in Figure 7. Our volumetric approach especially fits for such models, since it decouples the deformation from shape representation of the embedded model. Self-intersections, disconnected components, and non-manifold structures are handled easily.

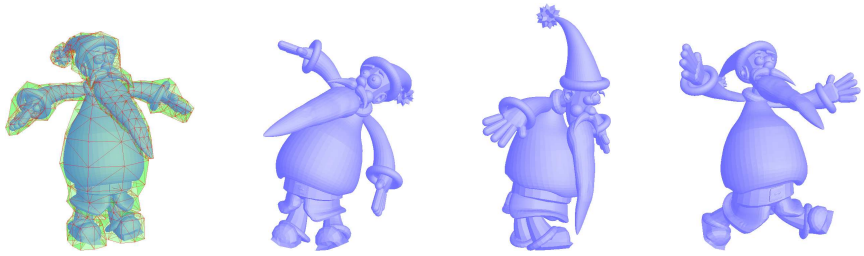
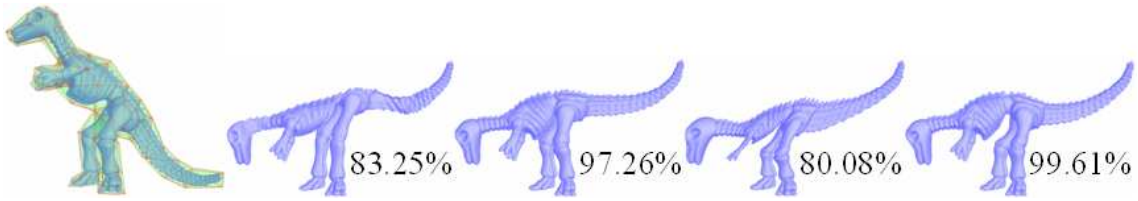


Figure 7: Deformation results of Santa with multiple parts and non-manifold structures. The leftmost shows the original Santa and its tetrahedral control mesh, the others are our results.



(a) (b) (c) (d) (e)

Figure 8: Comparison between different deformation methods. (a) original Dinosaur and its control mesh, (b) result of Lipman et al. [8], (c) result of Sorkine et al. [17], (d) result of Huang et al. [1], (e) our result. The numbers show the relative volume before and after deformation.

Figure 8 demonstrates a comparison between different deformation methods. Lipman et al. [8] and Sorkine et al. [17] are surface-based. Since [8] adopts uniform Laplacian without considering the irregularity of Dinosaur, apparent detail distortions and volume shrinkage appear. [17] rigidly deforms each edge of the mesh with an iterative framework, these undesirable artifacts can be avoided. Nevertheless, directly deforming 3D surface may suffer from slow convergence (the waist of Dinosaur). In [1], Huang et al. also present a volumetric method. However, it doesn't maintain the rigidity of control mesh during deformation, thus exhibits serious shape degenerations. By accounting for the rigidity energy, our method obtains a visual-pleasing result.

As to the two-phase deformation framework, solving sparse linear systems is the most time-consuming part. Because coefficient matrices are constant depend on the original control mesh, we can precompute their factorizations, then perform only back-substitutions during deformation. Table 1 shows the geometry data and time for deformation examples.

Deformation Transfer By setting the volumetric correspondence of the source and target tetrahedral control meshes, the deformation of the source model can be transferred to the target model. In Figure 4, Camel poses are used to generate a running Horse. Various Cat poses are retargeted to Lion in Figure 9, and the volumetric correspondence is established with only 15 markers.

Figure 10 gives a comparison with Sumner et al. [20]. Although the transferred results look similar, according to local zoom of anther view, our result preserves surface details better. Because of the volumetric nature of our transfer algorithm, it works well for models

Table 1: Performance data measured in milliseconds for interactive deformation. PRE: time for precomputation including energy discretization and matrix factorization. DEF: time for deformation including rotation estimation, back-substitutions and modified-BI.

Model	Num. of vertices	Num. of control vertices	Num. of tetrahedra	PRE	DEF
Armadillo	172,962	273	849	799.034	123.950
Asian dragon	249,934	173	473	406.546	119.222
Dragon	101,108	200	648	596.779	80.710
Santa	24,727	560	1616	1559.381	140.151
Dinosaur	56,194	204	567	509.845	63.751

with different genera (see Figure 11), a very difficult case for previous transfer algorithms to deal with. Table 2 illustrates the performance statistics for deformation transfer. All examples given in the paper run at interactive rates.

Conclusions and Future Work

Using coarse tetrahedral control meshes, this paper develops a unified framework to discuss two important editing operations: interactive deformation and deformation transfer.

A new rigidity energy is proposed to preserve geometric details and prevent unreasonable volumetric distortions even under large deformations. In addition, a dynamic refinement scheme based on deformation error is introduced for further optimization.

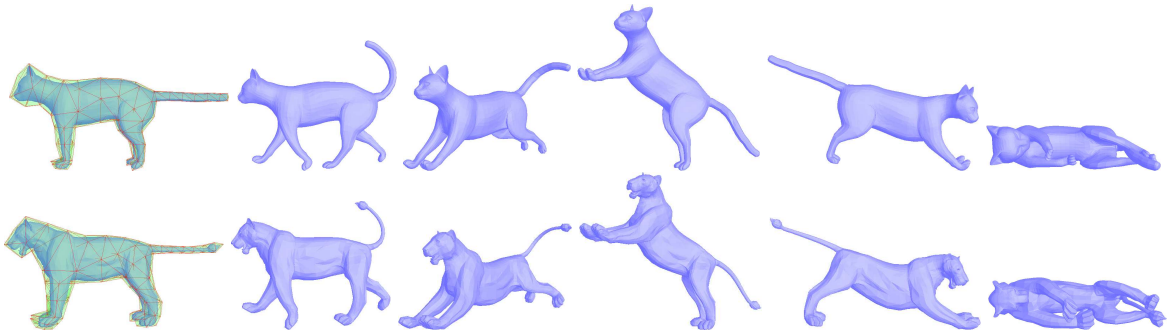
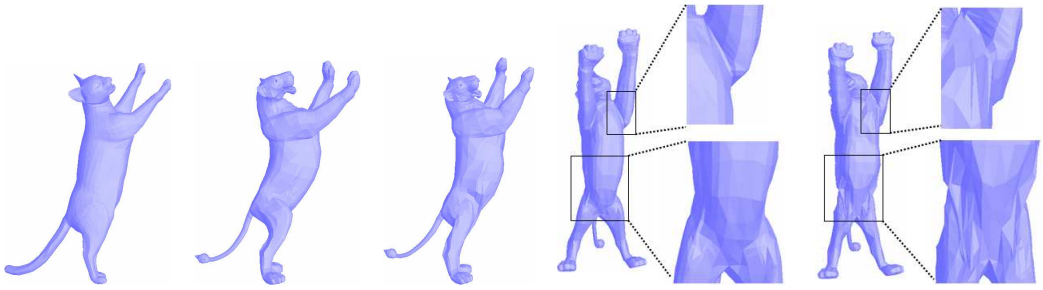


Figure 9: Deformation transfer (Cat to Lion). Various Cat poses are retargeted to Lion. The leftmost column shows the original models and their tetrahedral control meshes.



(a) (b) (c) (d) (e)

Figure 10: Comparison with Sumner et al. [20]. (a) a Cat pose, (b) our result, (c) result of Sumner et al. [20], (d) is in anther view of (b), (e) is in anther view of (c). Note that our algorithm preserves surface details better.

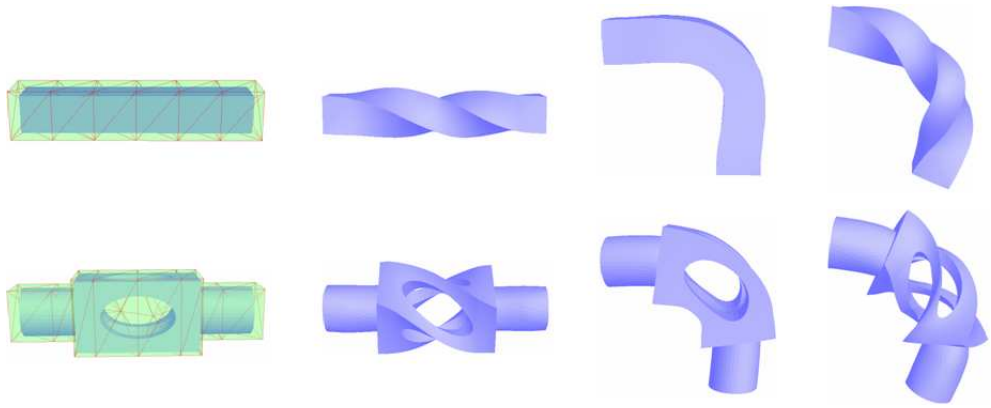


Figure 11: Deformation transfer between non-homeomorphous models. Twisting and bending deformations are transferred from Bar to Block. The leftmost column shows the original models and their tetrahedral control meshes.

Table 2: Performance data measured in milliseconds for deformation transfer. VP: time for setting volumetric correspondence. PRE: time for precomputation of fitting the deformed control mesh and deformation transfer optimization. FIT: time for fitting process. TO: time for transfer optimization and modified-BI.

Model	Num. of vertices	Num. of control vertices	Num. of tetrahedra	VP	PRE	FIT	TO
Camel Horse	21,887 8,431	467 512	1375 1463	67.895	2345.814	108.900	123.855
Cat Lion	7,207 5,000	200 209	522 567	28.240	759.190	37.442	45.464
Bar Block	23,402 11,371	35 57	78 147	5.906	629.264	5.402	15.989

Then based on this deformation scheme, a volumetric correspondence method is presented to perform deformation transfer task between the source and target tetrahedral control meshes, thus drastically reducing manual effort.

The volumetric nature of our approach allows it to handle various shape representations, and transfer deformations between non-homeomorphous models. These advantages make our framework conceptually intuitive and robust for practical applications.

Currently, our refinement scheme simply subdivides one tetrahedron to eight sub-tetrahedra, we will provide a more flexible way to deal with complicate cases. Our transfer algorithm works for models with similar poses. If they differ too much, the transferred results seem strange. Future work will also address performance improvements by implementing computations in GPU.

References

[1] J. Huang, L. Chen, X. Liu, and H. Bao. Efficient mesh deformation using tetrahedron control mesh. In *Proceedings of ACM symposium on Solid and physical modeling*, pages 241–247, 2008.

[2] T. Ju, S. Schaefer, and J. Warren. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.*, 24(3):561–566, 2005.

[3] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki. Harmonic coordinates for character articulation. *ACM Trans. Graph.*, 26(3):Article No. 71, 2007.

[4] Y. Lipman, D. Levin, and D. Cohen-Or. Green coordinates. *ACM Trans. Graph.*, 2008.

[5] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In *Proceedings of SIGGRAPH*, pages 325–334, 1999.

[6] M. Botsch and L. Kobbelt. Multiresolution surface representation based on displacement volumes. *Computer Graphics Forum*, 22(3):483–492, 2003.

[7] B. Sauvage, S. Hahmann, and G.-P. Bonneau. Volume preservation of multiresolution meshes. *Computer Graphics Forum*, 26(3):275–283, 2007.

[8] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl, and H.-P. Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, pages 181–190, 2004.

[9] O. Sorkine, Y. Lipman, D. Cohen-Or, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 179–188, 2004.

[10] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. SHUM. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.*, 23(3):644–651, 2004.

[11] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. SHUM. Large mesh deformation using the volumetric graph laplacian. *ACM Trans. Graph.*, 24(3):496–503, 2005.

[12] O. K.-C. Au, C.-L. Tai, L. Liu, and H. Fu. Dual laplacian editing for meshes. *IEEE Transaction on Visualization and Computer Graphics*, 12(3):386–395, 2006.

[13] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, and H.-Y. Shum. Subspace gradient domain mesh deformation. *ACM Trans. Graph.*, 25(3):1126–1134, 2006.

[14] Y. Lipman, D. Cohen-Or, R. Gal, and D. Levin. Volume and shape preservation via moving frame manipulation. *ACM Trans. Graph.*, 26(1):Article No. 5, 2007.

[15] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. Primo: coupled prisms for intuitive surface modeling. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 11–20, 2006.

[16] M. Botsch, M. Pauly, M. Wicke, and M. Gross. Adaptive space deformations based on rigid cells. *Computer Graphics Forum*, 26(3):339–347, 2007.

[17] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Proceedings of Eurographics Symposium on Geometry Processing*, pages 109–116, 2007.

[18] R.W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3):Article No. 80, 2007.

[19] A.R. River and D.L. James. Fastlsm: fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph.*, 26(3):Article No. 82, 2007.

[20] R.W. Sumner and Popović J. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, 2004.

[21] R. Zayer, C. Rössl, Z. Karni, and H.-P. Seidel. Harmonic guidance for surface deformation. *Computer Graphics Forum*, 24(3):601–609, 2005.

[22] Y. Chang, B. Chen, W. Luo, and J. Huang. Skeleton-driven animation transfer based on consistent volume parameterization. In *Proceedings of Computer Graphics International*, 2006.

[23] Y. Wang and T. Lee. Example-driven animation synthesis. *The Visual Computer*, 24(7):765–773, 2008.

[24] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Meshless deformations based on shape matching. *ACM Trans. Graph.*, 24(3):471–478, 2005.

[25] X. Shi, K. Zhou, Y. Tong, M. Desburn, H. Bao, and B. Guo. Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph.*, 26(3):Article No. 81, 2007.

[26] O. Sorkine and D. Cohen-Or. Least-squares meshes. In *Proceedings of the Shape Modeling International*, pages 191–199, 2004.

[27] Y. Tong, S. Lombeyda, A.N. Hirani, and M. Desbrun. Discrete multiscale vector field decomposition. *ACM Trans. Graph.*, 22(3):445–452, 2003.