EUROGRAPHICS 2009 / T. Akenine-Möller and M. Zwicker
(Guest Editors)

Volume 29 (2010), Number 2

Deformation Transfer to Multi-Component Objects

Kun Zhou[†] Weiwei Xu[‡] Yiying Tong[§] Mathieu Desbrun[¶]

[†] Zhejiang University [‡] Microsoft Research Asia [§] Michigan State University [¶] Caltech

Abstract

We present a simple and effective algorithm to transfer deformation between surface meshes with multiple components. The algorithm automatically computes spatial relationships between components of the target object, builds correspondences between source and target, and finally transfers deformation of the source onto the target while preserving cohesion between the target's components. We demonstrate the versatility of our approach on various complex models.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Geometry—Shape Deformation

1. Introduction

Deformation transfer [SP04] has proven to be a valuable technique for remapping rigid or non-rigid animation sequences from one geometry onto another. Despite its ability to transfer entire animation sequences from a surface mesh to another with only little user interaction, its most serious limitation lies in how similar (in shape and topology) the source and target meshes must be. This technique will simply fail if the source and target objects have drastically different topologies, *e.g.*, if the source is a single connected mesh while the target is an object containing multiple components (see Figure 1). However in real-life applications, characters or moving objects usually consist of multiple connected components, as pointed out in a recent paper by Ben-Chen et al. concurrent to our work [BCWG09]. They presented an efficient *space* deformation transfer technique to handle multi-component objects equipped with polyhedral cages. However, automatic cage generation methods often generate undesirable cages (*e.g.*, fused feet of humanoid models). Moreover, cage-based techniques can also run into numerical issues for thin-shell-like deformation as in Figure 3, where the cage would exhibit significant self-intersection and degeneracy.

In this paper, we remedy these issues by improving upon the surface-based deformation transfer algorithm. Our contribution stems from simple solutions to two main challenges: how to reliably establish correspondences between wildly different source and target meshes without heavy user interaction, and how to maintain the spatial distribution of the multiple components of the target throughout an animation sequence. The versatility of our method allows for a much wider range of applications, as we demonstrate through deformation transfer onto whimsical objects, and retargeting captured motion data onto multi-component targets.

1.1. Related Work

Deformation transfer [SP04] provides a simple and efficient way to remap animation sequences from one mesh onto another. This method first builds a correspondence map between the triangles of the source mesh and those of the target mesh by deforming the target into the reference pose of the source through energy minimization. Then an *affine transformation matrix* (or deformation gradient) is computed from each triangle in the source mesh to its counterpart in the deformed source mesh by appending an additional vertex to each triangle. Finally the transformation matrices are applied to the corresponding triangles in the target mesh, and a linear system (i.e., Poisson equation) is solved to generate the deformed target mesh whose deformation gradients (with respect to the reference target mesh) are as similar as possible to the deformation gradients of the source. The final process is also known as gradient-domain mesh deformation, which has gained a lot of attention in recent years [YZX*04, SCOL*04, BS08].

Since its introduction, the deformation transfer method has been improved and extended by many researchers. Instead of transferring all the gradients of the source deformation, Zayer et al. [ZRKS05] proposed to transfer gradients only at a few key points, and interpolate them to other points of the target with a harmonic function. Botsch et al. [BSPG06] proved that the same transfer results can be achieved without

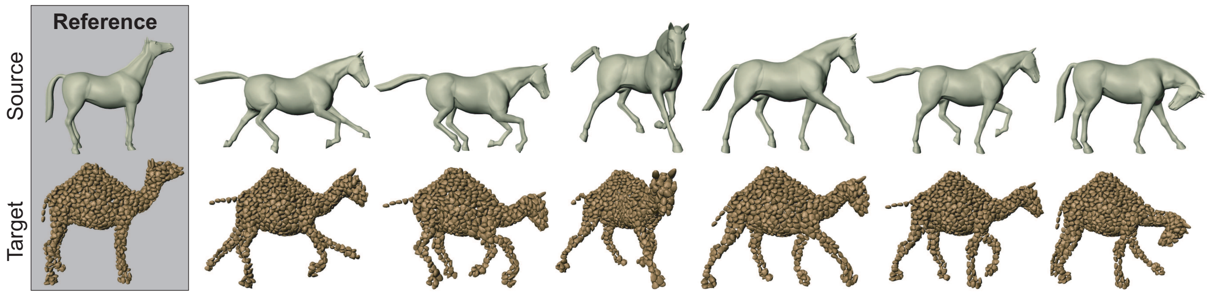

Figure 1: *Deformations of a single-component horse are transferred to a pebble camel made of 750 separate stones. Only 47 correspondences are specified. (The pebble camel was inspired by The Thing, a fictional character made of stones in the movie entitled "The Fantastic Four".)*

adding an additional vertex to each triangle, speeding up the linear system solve. While the original deformation transfer method requires similar reference poses between the source and target meshes, Lee et al. [LWC06] developed a method to remove this restriction. They segment the source and target into the same number of near-rigid components and adjust the orientation of the target components to be similar to the corresponding source components through rigid transformation. More recently, Baran et al. [BVGP09] proposed *semantic* deformation transfer to preserve the semantic characteristics of the motion instead of its literal deformation.

One serious limitation of Sumner and Popović's method is that it is restricted to single-component meshes. To remove this restriction, Ben-Chen et al. [BCWG09] proposed a *spatial* deformation transfer technique to handle multiple-component meshes and polygon soups. Their technique, however, requires to construct polyhedral cages for the objects. Moreover, some high-frequency details of the source deformation may be lost since the method projects the deformation into a low-dimensional linear subspace. Like Sumner and Popović's original approach, our method is surface-based, and thus does not suffer from these issues.

1.2. Algorithm Overview

Our goal is to extend Sumner and Popović's method to multi-component meshes. We adopt their approach for the computation of the per-triangle deformation. However, the original method requires a manual specification of a global boundary condition, *i.e.*, an anchor point for each topological component of the mesh to assemble the deformations. Handling objects with multiple parts could thus not be handled for a series of reasons: (1) providing an anchor for each separate component of the target object is difficult; (2) preservation of spatial relationships between components is not addressed; (3) establishing correspondence between the source mesh and the target mesh requires tedious manual work to specify enough marker pairs on each component. To circumvent these issues, we will show that one can employ simple graph theoretical techniques, along with recent developments in mesh deformation that locally preserve shape as best as possible [BS08].

Our algorithm first computes a set of closest vertex pairs to build a graph that encodes the spatial relationship between components. Then both the correspondence algorithm and the deformation transfer algorithm in [SP04] are adapted to take into account the graph connectivity. First, when establishing correspondences between the source and target, we add an energy term to make the components without any user-specified markers deform with the components that have makers. This greatly reduce the manual work to specify markers for each separate component. Note that, unlike in [SP04], where every triangle of the target has a corresponding triangle in the source, not all target triangles may have correspondences in our case. Some components of the target may not have any corresponding triangles in the source. Such components are called *orphan components*.

Second, when deforming the target to match the source deformation gradient, we add two new energy terms as soft constraints. The first term tries to preserve the lengths of the graph edges so that the spatial relationship between components is maintained. It also removes the need to provide an anchor for each separate component of the target object. The second term is based on Laplacian coordinates [SCOL*04] and tries to preserve the surface details of the orphan components in the target. Since the two newly added terms are not in quadratic form, the total energy cannot be minimized using a linear solver. Instead, an iterative Gaussian-Newton method is employed – we approximate the two terms as quadratic functions and only need to solve a linear system at each iteration. This optimization scheme works efficiently for all of our experimental data.

One may think that the above two energy terms are unnecessary, and by interpolating the deformation of orphan components from nearby components, the desired deformation can be computed using a linear solver. The assumption of this linear approach (i.e., the closest triangle pairs of nearby components have similar deformations), however, is not true in many cases. For example, when the mechanical-like horse shown in Figure 3 is running, two components near to a joint can have different rotations. Enforcing transformation smoothness between the closest triangle pairs may introduce undesirable deformation artifacts (see Figure 3).

2. Algorithm

Given a pair of source objects, $\mathbf{S}$ and $\tilde{\mathbf{S}}$, and a target object $\mathbf{T}$ consisting of multiple disconnected components ($\mathbf{T} = \{T_1, T_2, ...\}$), our goal here is to generate a new object $\tilde{\mathbf{T}}$ such that the deformation between $\mathbf{T}$ and $\tilde{\mathbf{T}}$ is analogous to the deformation between $\mathbf{S}$ and $\tilde{\mathbf{S}}$. Note that the source object may also consist of multiple components. For clarity, we will denote by $\mathbf{u}_i$ a vertex of $\mathbf{S}$ and by $\tilde{\mathbf{u}}_i$ its corresponding vertex in $\tilde{\mathbf{S}}$, while $\mathbf{v}_i$ will refer to a vertex of $\mathbf{T}$ and $\tilde{\mathbf{v}}_i$ its associated vertex in $\tilde{\mathbf{T}}$ that we will solve for. We will also sometimes indicate with a superscript that a vertex belongs to one of the components of $\mathbf{T}$; *e.g.*. $\mathbf{v}_i^a$ and $\mathbf{v}_i$ will refer to the same vertex, but the superscript stresses the fact that $\mathbf{v}_i$ belongs to the component T_a.

Our approach starts by adding vertices to the source objects. Following [SP04], for each triangle of $\mathbf{S}$ (resp., $\tilde{\mathbf{S}}$) with three vertices $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$ (resp., $\{\tilde{\mathbf{u}}_1, \tilde{\mathbf{u}}_2, \tilde{\mathbf{u}}_3\}$), we add a fourth vertex $\mathbf{u}_4$ (resp., $\tilde{\mathbf{u}}_4$) in the direction of the outward normal of the triangle, at a distance from the triangle proportional to the square root of the area. This procedure is designed to make the stretch in the normal direction the geometric average of the principal stretches in the plane. The affine transformation of the source triangle, mapping $\mathbf{u}_i$ to $\tilde{\mathbf{u}}_i$ for $i = 1 \ldots 3$, is computed as:

$$\mathbf{F} = \tilde{\mathbf{G}}\mathbf{G}^{-1}, \tag{1}$$

where

$$\begin{aligned} \mathbf{G} &= [\mathbf{u}_2 - \mathbf{u}_1 \quad \mathbf{u}_3 - \mathbf{u}_1 \quad \mathbf{u}_4 - \mathbf{u}_1] \\ \tilde{\mathbf{G}} &= [\tilde{\mathbf{u}}_2 - \tilde{\mathbf{u}}_1 \quad \tilde{\mathbf{u}}_3 - \tilde{\mathbf{u}}_1 \quad \tilde{\mathbf{u}}_4 - \tilde{\mathbf{u}}_1]. \end{aligned}$$

The algorithm then proceeds as follows:

1. *Computation of Proximity Pairs between Components*: Since $\mathbf{T}$ consists of multiple disconnected components, we start by encoding the spatial relationship between components through a set of *proximity pairs*. For each pair of components (T_a, T_b), we will define as P_{ab} a selected set of *vertex index pairs* that represent key spatial adjacencies that subsequent processing will try to preserve during transfer. Note that P_{ab} may be empty if the associated components are too far away from each other. The computation of P_{ab} will be described in Section 2.1.

2. *Establishment of Correspondences*: We assume that the user has provided a *small* set of markers that indicate point-to-point correspondences between meshes $\mathbf{S}$ and $\mathbf{T}$. From these markers, we will deduce correspondences between source triangles and target triangles through a set M:

$$M = \{(s_1, t_1), (s_2, t_2), ..., (s_{|M|}, t_{|M|})\}, \tag{2}$$

where a pair (s_m, t_m) indicates that the target triangle t_m should deform like the source triangle s_m. The computation of M will be explained in Section 2.2. Unlike in [SP04], where every triangle of the target object has a corresponding triangle in the source object, not all target triangles may have correspondences. If the source and target objects have drastically different topologies, enforcing a correspondence to a triangle on the source for each triangle of the target and transferring the deformation will generate unintuitive and unappealing results (see Figure 3). We denote by H the set of "orphan" target triangles (*i.e.*, the ones that do *not* have correspondences). Note that some components in $\mathbf{T}$ may not contain any corresponding triangles in M. We denote by T_H the set of these "orphan" components without corresponding triangles.

3. *Deformation Transfer*: Finally, deformation is transferred through an energy minimization that accounts for the presence of multiple components. Section 2.3 reviews this final process.

We now provide details on the successive steps of our deformation transfer approach.

2.1. Proximity of Components

The spatial relationship between the components of the target mesh, encoded as *proximity pairs*, can be established through the following steps:

- For every pair of components (T_a, T_b), find the shortest distance $d_{a,b}$ between them as well as the pair of vertices $(v_{i_1}^a, v_{j_1}^b)$ corresponding to this shortest distance:

$$d_{a,b} := \|\mathbf{v}_{i_1}^a - \mathbf{v}_{j_1}^b\| = \min_{v_i \in T_a, v_j \in T_b} \|\mathbf{v}_i - \mathbf{v}_j\|.$$

- Build a complete graph G with the components as its nodes, where edges are weighted by the shortest distances computed above.

- Compute a minimal spanning tree K (*i.e.*, the minimum weight subgraph) of the graph.

- For each component T_a, compute its largest distance d_a from each of its adjacent components in K:

$$d_a = \max_{\text{edge } (a,b) \in K} d_{a,b}$$

- For every pair of components T_a and T_b *not* connected by an edge in the subgraph K, if $d_{a,b} \le d_a + \varepsilon_a$ and $d_{a,b} \le d_b + \varepsilon_b$, *connect* T_a and T_b by adding an edge in K between them. The user-specified thresholds ε_a and ε_b define a notion of "close" proximity, controlling how cohesive the target must remain. In our implementation, ε_a

(resp., ε_b) is computed as the maximal edge length of the component T_a (resp., T_b) times 1.5.

The proximity pairs are then easily defined: for two components T_a and T_b, $P_{a,b}$ is set to nil if these components are not connected by an edge in the subgraph K. Otherwise, we set $P_{a,b}$ to be a set initialized with the pair $(v^a_{i_1}, v^b_{j_1})$; we further add to $P_{a,b}$ any vertex pair $(v^a_{i_k}, v^b_{j_k})$ such that $\|\mathbf{v}^a_{i_k} - \mathbf{v}^b_{j_k}\| < d_{a,b} + \min\{\varepsilon_a, \varepsilon_b\}$.

This simple choice of using a few distance-based vertex pairs to encode proximity provides robustness to all sorts of targets while limiting undesirable relative rotations between nearby components, as we will demonstrate in Section 3.

2.2. Establishing Correspondences

Finding correspondences between the source and target objects is achieved by deforming the target $\mathbf{T}$ into the static pose of the source $\mathbf{S}$ in a fashion nearly identical to the original deformation transfer approach [SP04]. The deformed vertices of $\mathbf{T}$ that we need to solve for will be called $\mathbf{x}_i$'s, and are only temporarily used in this procedure to establish correspondences.

The user controls the deformation by specifying a set of pairs of source and target vertex indices. Each pair indicates that the target vertex, after deformation, should match the location of the source vertex. Then an iterated closest point algorithm is performed. At each iteration, the target is deformed through an energy minimization, and the set of closest valid vertex-point pairs is updated.

Correspondence Energy We compute the deformed vertices $\mathbf{X} = [\mathbf{x}_1^t \mathbf{x}_2^t \ \ldots]^t$ by minimizing the following energy:

$$\begin{aligned} E &= w_S E_S + w_I E_I + w_C E_C + E_R \\ \text{s.t.} \quad & \mathbf{x}_{m_k} = \mathbf{m}_k, \quad k \in 1 \ldots m \end{aligned} \tag{3}$$

where m_k is the vertex index on the target mesh for marker k, and $\mathbf{m}_k$ is the corresponding position of marker k on the source object. The deformation smoothness term E_S, the deformation identity term E_I, the closest valid point term E_C and the choice of weights are all identical to [SP04]: our only change to the correspondence process lie in the last energy term E_R. The purpose of this added term is to make the components without any user-specified markers *deform along* with the components that do have markers, thereby minimizing the need for user-defined correspondences. The E_R energy is computed as follows:

- For each component pair (T_a, T_b), we find a pair of triangles $(t^a_{i_k}, t^b_{j_k})$ for each vertex pair $(v^a_{i_k}, v^b_{j_k})$ in $P_{a,b}$, such that $t^a_{i_k} \in \mathcal{N}(v^a_{i_k}), t^b_{j_k} \in \mathcal{N}(v^b_{j_k})$ ($\mathcal{N}(.)$ refers to the usual topological one-ring), and that the triangles have the largest separation along the normal directions. More precisely, the pair is defined as:

$$\arg\max_{t^a_{i_k}, t^b_{j_k}} |\mathbf{h} \cdot \mathbf{n}_{t^a_{i_k}}| + |\mathbf{h} \cdot \mathbf{n}_{t^b_{j_k}}|,$$

  where $\mathbf{n}_t$ is the normal of triangle t, and $\mathbf{h}$ is the vector between the barycenters of the two triangles, $t^a_{i_k}$ and $t^b_{j_k}$.
- For each triangle $t^a_{i_k}$ we found above, we construct a tetrahedron by using one of the three vertices of $t^b_{j_k}$ as the fourth vertex. Similarly, we construct a tetrahedron for triangle $t^b_{j_k}$. However, if triangles $t^a_{i_k}$ and $t^b_{j_k}$ are approximately coplanar in the rest pose of the target object, the two constructed tetrahedra will be degenerate. In this case we simply use $t^a_{i_k}$'s fourth vertex (that was added in the direction perpendicular to the triangle, see Section 2) as the fourth vertex of both tetrahedra.
- The energy E_R is then defined as:

$$E_R = \sum_{a,b} \sum_{k=1}^{|P_{a,b}|} \left(w_S \|\hat{\mathbf{F}}_{t^a_{i_k}} - \hat{\mathbf{F}}_{t^b_{j_k}}\|^2 + w_I \|\hat{\mathbf{F}}_{t^a_{i_k}} - \mathbf{I}\|^2 + w_I \|\hat{\mathbf{F}}_{t^b_{j_k}} - \mathbf{I}\|^2 \right), \tag{4}$$

  where $\hat{\mathbf{F}}_{t^a_{i_k}}$ indicates the transformation of the newly constructed tetrahedron based on $t^a_{i_k}$.

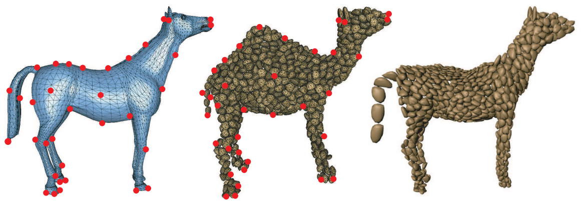
Figure 2: *Finding correspondences through deforming the target into the source. From left to right are the source, the target, and the deformed target object. User-specified marker points are shown in red.*

Note that our goal is to deform the target into the source to establish correspondences. After deformation, the shape of every component as well as the edge length between components may have changed greatly. We thus cannot base our energy on edge length constraint or on the Laplacian constraint, both of which are described in Section 2.3 (as E_3 and E_4). Our solution, implemented through E_R, of adding tetrahedra between two components allows for affine transformations of components, and is therefore more effective.

Solving for Correspondences The correspondence energy minimization is achieved via the same two-phase method described in [SP04], consisting in successive linear solves. At each iteration, after the target object is deformed we also need to update the set of the closest valid vertex-point pairs. For each vertex of the deformed target object at the current iteration, we find its closest point on the source object. If the distance between vertex and point is less than a user-specified threshold and the difference between the vertex's normal and the point's normal is less than 90 degree, we

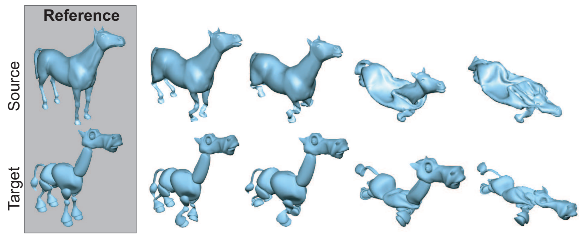

Figure 3: *A mechanical-like horse collapsing.*

add the vertex and point to the valid set. The whole process converges fast and reaches a minimum after a few iterations (see statistics in Section 3). We then compute the triangle correspondences between the source and the deformed target triangles. For each deformed target triangle, if one or more of its three vertices are among the markers or in the set of the closest vertex-point pairs, it will not be assigned any corresponding source triangle. Otherwise, we label its corresponding source triangle as the one with the *smallest distance* between the barycenters of the two triangles.

2.3. Deformation Transfer

We finally solve for deformed vertex positions $\tilde{\mathbf{V}} = [\tilde{\mathbf{v}}_1^t \tilde{\mathbf{v}}_2^t \ldots]^t$ by minimizing the following energy:

$$E = w_1 E_1 + w_2 E_2 + w_3 E_3 + w_4 E_4. \qquad (5)$$

Each term (explained next) serves a different purpose, while the weights offer control on the overall deformation transfer behavior. We use weights $w_1 = w_2 = w_4 = 1.0$ and $w_3 = 0.1$, for all the examples in this paper.

The E_1 term measures the difference between the source and target transformations (defined in Eq. (1)) :

$$E_1 = \sum_{i=1}^{|M|} \|\mathbf{F}_{s_i} - \mathbf{F}_{t_i}\|^2. \qquad (6)$$

For each triangles in H (*i.e.*, with no corresponding triangle in the source object), E_2 acts as a regularization, forcing the transformation of this orphan triangle to be close to the transformations of its adjacent triangles (where adjacency is, again, defined as its topological one-ring):

$$E_2 = \sum_{t_i \in H} \sum_{t_j \in \mathcal{N}(t_i)} \|\mathbf{F}_{t_i} - \mathbf{F}_{t_j}\|^2. \qquad (7)$$

The E_3 term helps maintain the spatial relationship between components by preserving the edge lengths of the vertex pairs in each $P_{i,j}$:

$$E_3 = \sum_{\substack{T_a, T_b \in T \\ a \neq b}} \sum_{k=1}^{|P_{a,b}|} (\|\tilde{\mathbf{v}}_{i_k}^a - \tilde{\mathbf{v}}_{j_k}^b\| - \|\mathbf{v}_{i_k}^a - \mathbf{v}_{j_k}^b\|)^2. \qquad (8)$$

Note that $\|\mathbf{v}_{i_k}^a - \mathbf{v}_{j_k}^b\|$ is computed on the rest pose of the target object and remains constant. Finally, the components in T_H do not have corresponding triangles in the source object.

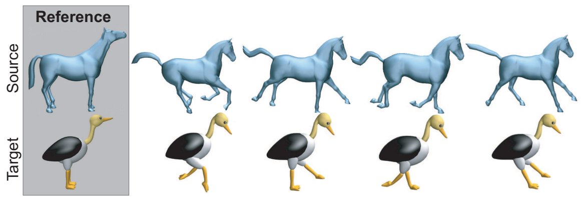

Figure 4: *A 9-component bird running like a horse.*

Model	Components	Vertices	Triangles
Horse	1	8,431	16,843
Woman (source)	1	9,971	19,938
Car (source)	261	26,362	47,315
Mechanical Horse	21	10,209	20,334
Pebble Camel	750	51,000	99,000
Bird	9	6,712	13,388
Woman (target)	4	12,594	25,704
Car (target)	40	28,885	57,332

Table 1: *Number of components, vertices and triangles for the models used in the paper.*

Thus, we use Laplacian coordinates [SCOL*04] to *preserve the surface details* of these components by adding a fourth energy:

$$E_4 = \sum_{T_a \in T_H} \|\mathbf{L}_{T_a} \tilde{\mathbf{V}}_{T_a} - \hat{\delta}(\tilde{\mathbf{V}}_{T_a})\|^2, \qquad (9)$$

where $\mathbf{L}_{T_a}$ is the Laplacian operator matrix computed on the mesh T_a, $\tilde{\mathbf{V}}_{T_a}$ is a sub-array of $\tilde{\mathbf{V}}$ containing only the vertex positions of component T_a, and the Laplacian coordinates $\hat{\delta}(\tilde{\mathbf{V}}_{T_a})$ are defined as in [HSL*06]:

$$\hat{\delta}(\tilde{\mathbf{V}}_{T_a}) = \frac{\mathbf{L}_{T_a} \tilde{\mathbf{V}}_{T_a}}{\|\mathbf{L}_{T_a} \tilde{\mathbf{V}}_{T_a}\|} \|\mathbf{L}_{T_a} \mathbf{V}_{T_a}\|.$$

The total energy E is therefore a nonlinear function of $\tilde{\mathbf{V}}$, but can be efficiently minimized using an iterative Gauss-Newton method as advocated in [SZT*07]: while E_1 and E_2 are quadratic functions of $\tilde{\mathbf{V}}$, repeated quadratic approximations of E_3 and E_4 lead to a linear solve for every Gauss-Newton iteration. Note that the optimization problem requires an additional boundary condition to determine the global translation: this is easily fixed by setting the position for a target vertex, or by specifying another positional constraint such as foot placement.

Note that all energy terms in Eq. (5) were carefully designed, and proved to be necessary in our experiments. In particular, the smoothness term E_2 is used to propagate the transformations of the triangles with correspondences to those orphan triangles without correspondences. The Laplacian term E_4 is used to preserve the surface details of orphan components. Omitting E_2 would leave the transformation matrices of the orphan triangles discontinuous. Thus, the positions of these vertices could not be determined and the orphan triangles would not be deformed accordingly without E_2. Omitting E_4

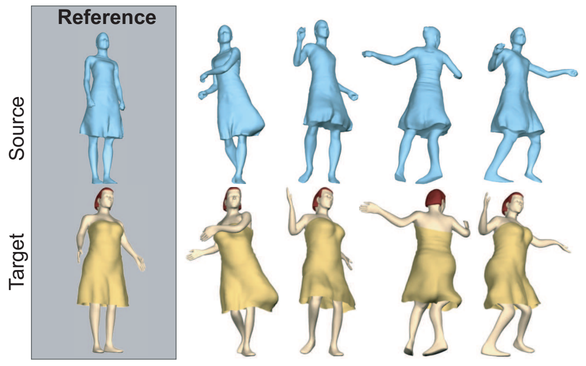

Figure 5: *A woman in a dress follows the moves of a dancer.*

would leave orphan components underconstrained. The positions of orphan components could thus not be determined.

3. Results

We have implemented our deformation transfer algorithm on an Intel Xeon 3.7GHz workstation. We selected a number of examples to demonstrate the versatility of the resulting algorithm. We provide statistics for the models presented in this paper in Table 1 and Table 2, including timings and number of markers needed.

Whimsical Objects The first example shows a deformation of a single-component horse transferred to a robot-looking horse which has 21 components (see Figure 3). Both global and local deformations are nicely reproduced on the new model. Moreover, all components of the model deform consistently and their spatial relationship is well maintained. Note that we constrained a marker vertex on one of the target's feet components to match its correspondence marker on the source. In Figure 4, we transfer the deformations of the horse to a 9-component bird. Although every component intersects with its adjacent components, our algorithm robustly handles the deformation transfer, producing very reasonable results.

Objects with Many Components To test the robustness of our correspondence and transfer algorithm further, we try to transfer the deformations of a horse onto a pebble camel made of 750 separate stones (see Figure 1). In this case, only 47 correspondence markers are specified. Although most components do not have markers, our correspondence algorithm handles them reliably via the extra tetrahedra (added in the energy E_R), and nicely deforms the camel into the horse, establishing satisfactory correspondences. The spatial relationships between components are very well preserved while the deformations are faithfully transferred.

Human body & Clothes In [VBMP08], complex captured data are encoded as a time-varying single-component mesh. However, transferring deformation from such captured data to other meshes will often involve multiple-component targets: clothes and body meshes (as in Figure 5) are typically given as *separate* meshes. We show that our approach can transfer animations from multiview silhouettes to multi-component models created using commercial modeling softwares. Note that in some regions of the target model, the body mesh is very close to the cloth mesh; they both are deformed into the source mesh and find correct correspondences. As a result, these regions of the body are deformed along with the cloth.

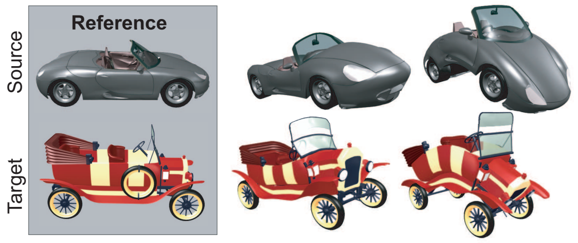

Figure 6: *Deformation transfer of multi-component objects.*

Example	# Markers	Build Corresp.	Transfer
Horse/Mec. Horse	43	8.15s	0.424s
Horse/Bird	19	5.65s	0.439s
Horse/Pebble Camel	47	54.56s	14.72s
Woman	78	14.87s	1.83s
Car	70	32.86s	8.09s

Table 2: *The number of correspondence markers and the timing results (in seconds) for building correspondence and transferring a single frame.*

Our last example shows that our algorithm can easily handle deformation transfer from one multi-component object to another. As shown in Figure 6, the deformation of a 261-component sports car is transferred to a 40-component vintage car model. Note that the algorithm of Section 2 naturally works for multi-component source objects *as is*.

4. Conclusion

We have generalized the deformation transfer technique of [SP04] to handle complex models consisting of arbitrarily many components. We proposed simple methods to establish spatial relationship between components, build correspondences between models with different topologies, and transfer deformation while preserving pairwise component proximity. The resulting algorithm is straightforward to implement.

Although our algorithm works well even in the cloth examples, we cannot guarantee collision-free deformation transfer (see Figure 4). This may be, in certain applications, a serious limitation. For future work, we believe that parts of our algorithm can be easily adopted by as-rigid-as-possible shape interpolation algorithms [ACOL00] to handle multi-component objects. We are also interested in defin-

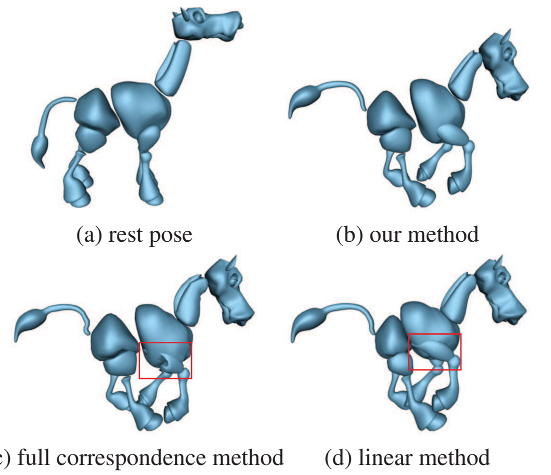

Figure 7: *Comparison on the horse sequence. (a) Rest pose. (b) Transfer result using our method. (c) Transfer result by enforcing a correspondence between a triangle on the source for* each *triangle of the target. Note the artifacts at the front legs of the horse. (d) Transfer result using a linear method, i.e., without the two terms E_3 and E_4 in Equation (5) and by imposing smoothness constraints on paired tetrahedra found as described in Section 2.2. As closest triangle pairs get different transformations from the source, this method introduces undesirable deformations. The relative distance between components are not well maintained either.*

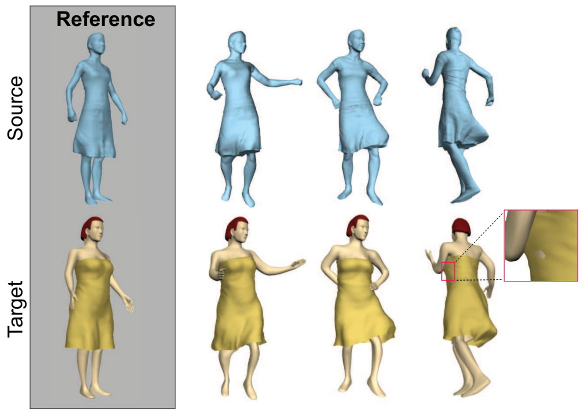

Figure 8: *A self-collision example. The samba deformation sequence in [Vlasic et al. 2008] is transferred to the same woman model in Figure 5. We only used 63 markers (compared to the 78 markers used in Figure 5). Self-collisions occur in a few frames of the transfer result.*

ing and preserving other kinds of spatial relationship between components during deformation transfer; currently we only use edge lengths between vertex pairs, which is inappropriate if the relative distance between components in the source model are supposed to change during deformation.Therefore, extensions could be proven useful.

Acknowledgements We would like to thank the anonymous reviewers for their helpful comments. This research was partially funded by the NSFC (No. 60825201), the 973 program of China (No. 2009CB320801), the NSF (CCF-0811313/0811373, and CMMI-0757106/0757123), and the Open Project Program of the State Key Lab of CAD&CG, Zhejiang University.

References

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-rigid-as-possible shape interpolation. In *ACM SIGGRAPH* (2000), pp. 157–164. 6

[BCWG09] BEN-CHEN M., WEBER O., GOTSMAN C.: Spatial deformation transfer. In *Proceedings of SCA'09* (2009). 1, 2

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE TVCG 14*, 1 (2008), 213–230. 1, 2

[BSPG06] BOTSCH M., SUMNER R., PAULY M., GROSS M.: Deformation transfer for detail-preserving surface editing. In *Vision, Modeling & Visualization* (2006), pp. 357–364. 1

[BVGP09] BARAN I., VLASIC D., GRINSPUN E., POPOVIĆ J.: Semantic deformation transfer. *ACM TOG 28*, 3 (2009), 36. 2

[HSL*06] HUANG J., SHI X., LIU X., ZHOU K., WEI L.-Y., TENG S.-H., BAO H., GUO B., SHUM H.-Y.: Subspace gradient domain mesh deformation. *ACM TOG 25*, 3 (2006), 1126–1134. 5

[LWC06] LEE T.-Y., WANG Y.-S., CHEN T.-G.: Segmenting a deforming mesh into near-rigid components. *The Visual Computer 22*, 9 (2006), 729–739. 2

[SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Proceedings of SGP'04* (2004), pp. 175–184. 1, 2, 5

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM TOG 23*, 3 (2004), 399–405. 1, 2, 3, 4, 6

[SZT*07] SHI X., ZHOU K., TONG Y., DESBRUN M., BAO H., GUO B.: Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM TOG 26*, 3 (2007), 81. 5

[VBMP08] VLASIC D., BARAN I., MATUSIK W., POPOVIĆ J.: Articulated mesh animation from multi-view silhouettes. *ACM TOG 27*, 3 (2008), 97. 6

[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 TOG 23*, 3 (2004), 644–651. 1

[ZRKS05] ZAYER R., RRÖSSL C., KARNI Z., SEIDEL H.-P.: Harmonic guidance for surface deformation. *Computer Graphics Forum 24*, 3 (2005), 601–609. 1