*To appear in SIGGRAPH 2004.*

# Deformation Transfer for Triangle Meshes

Robert W. Sumner  Jovan Popović

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

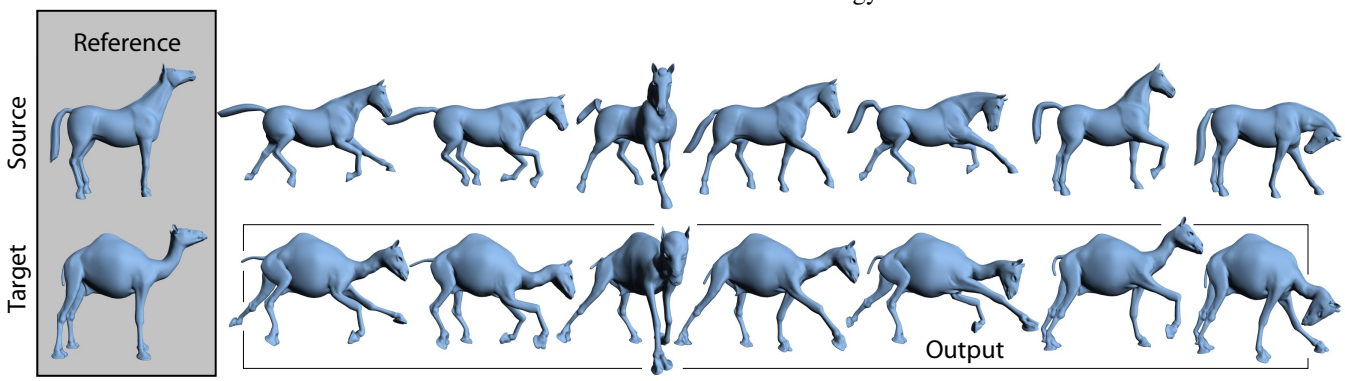


Figure 1: Deformation transfer copies the deformations exhibited by a source mesh onto a different target mesh. In this example, deformations of the reference horse mesh are transfered to the reference camel, generating seven new camel poses. Both gross skeletal changes as well as more subtle skin deformations are successfully reproduced.

## Abstract

Deformation transfer applies the deformation exhibited by a source triangle mesh onto a different target triangle mesh. Our approach is general and does not require the source and target to share the same number of vertices or triangles, or to have identical connectivity. The user builds a correspondence map between the triangles of the source and those of the target by specifying a small set of vertex markers. Deformation transfer computes the set of transformations induced by the deformation of the source mesh, maps the transformations through the correspondence from the source to the target, and solves an optimization problem to consistently apply the transformations to the target shape. The resulting system of linear equations can be factored once, after which transferring a new deformation to the target mesh requires only a backsubstitution step. Global properties such as foot placement can be achieved by constraining vertex positions. We demonstrate our method by retargeting full body key poses, applying scanned facial deformations onto a digital character, and remapping rigid and non-rigid animation sequences from one mesh onto another.

**CR Categories:** I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Hierarchy and geometric transformations; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

**Keywords:** Deformations, Correspondence, Animation

**Authors' contact:**
The Stata Center, 32 Vassar Street, Cambridge, MA 02139
sumner@csail.mit.edu
jovan@csail.mit.edu

## 1 Introduction

Mesh deformation plays a central role in computer modeling and animation. Artists hand-sculpt facial expressions and stylized body shapes. They assemble procedural deformations and may use complex musculature simulations to deform a character's skin. Despite the tremendous amount of artistry, skill, and time dedicated to crafting deformations, there are few techniques to help with reuse. In order to reuse a deformation created for one shape to deform another, the specific parameters that control the deformation must be adapted to the new shape. In many cases, adapting these parameters is just as time consuming as starting from scratch. Although special purpose adaption methods exist, the problem is compounded in the common case where many different deformation techniques are used in tandem. An automatic adaption method designed for one type of deformation may fail in the presence of others. Furthermore, any hand-sculpted alterations will be lost. As a result, the work spent designing a deformation typically cannot be reused after its planned application.

Our research amends this problem by automatically copying deformations from one mesh onto another. This deformation transfer technique is our central research contribution. We use a general approach that requires no knowledge of the actual method used to deform the original shape. Our technique is purely mesh-based and does not require the two meshes to share the same number of vertices or triangles, or to have identical connectivity. However, our algorithm is designed for the case where there is a clear semantic correspondence between the two meshes indicating which parts of the source and target should deform similarly. Our system can transfer hand-sculpted alterations as well as deformations resulting from arbitrarily complex procedural or simulation based methods. Figure 1 demonstrates our method used to transfer full body deformations of a horse mesh onto a camel. The camel mesh was never articulated and the resulting camel deformations are completely derived from the source mesh using deformation transfer.

With the aid of a correspondence tool, the user supplies a mapping between the triangles of the source and those of the target. For each triangle of the source mesh, our method computes an affine transformation that takes the triangle from its original position to its deformed position. These affine transformations, together with the correspondence, specify the ideal change in orientation, scale,

and skew of each triangle of the target shape. However, these ideal transformations will not, in general, be consistent with respect to one another: applied directly without modification, the transformations would not preserve the connectedness of the target mesh. Therefore, to find the deformed target shape, deformation transfer solves an optimization problem such that the ideal changes are matched as closely as possible while maintaining consistency. The retargeting process is numerically efficient, as the system of linear equations for a source/target pair can be factored and stored in a precomputation step. Transferring a new deformation from the source to the target requires only performing backsubstitution with the stored factorization. Global properties such as foot placement can be achieved using positional vertex constraints.

## 2 Background

Deformation transfer is a generalization of the concept introduced by expression cloning, which transfers facial expressions from one face mesh to another [Noh and Neumann 2001]. In this approach, each expression is encoded with vertex displacements that define the differences between the reference face and the expression face. Expression cloning uses heuristics designed to adapt the direction and scale of displacement vectors to account for faces of differing shape and proportions. This representation and adaptation technique is specialized for the deformations that arise in facial expressions. Our method transfers arbitrary nonlinear deformations by computing an optimal global deformation of the target shape.

One way to represent such a global deformation is with a free-form deformation [Sederberg and Parry 1986] in which a lattice of control points induces a deformation on the enclosed space. With this or a similar representation, any target mesh can be deformed with ease by applying the global deformation to every mesh vertex. The Inkwell 2D system uses precisely this strategy to animate different 2D characters with the same set of hand-animated Coons patches [Litwinowicz 1991]. However, this approach is harder to generalize when the source deformation is not initially described by a free-form deformation or a similar representation. In these cases, the method must infer both the structure of the control lattice and the position of its control points, or, less optimally, solve for the control points of a specific lattice structure [Hsu et al. 1992]. A fixed lattice structure is not optimal, because a reasonably sized lattice cannot express arbitrary nonlinear deformations of vertices for every target mesh [Singh and Fiume 1998].

Deformation transfer resolves this problem by using locally specified deformations [Barr 1984], which can define any global nonlinear deformation of mesh vertices. This approach extends the ideas from Alexa, Cohen-Or, and Levin's [2000] shape interpolation technique which maximizes the rigidity of a blended shape by computing the optimal deformation of its interior. We show that the interior of the target mesh need not be considered for transfer of mesh deformation. Our boundary formulation has tremendous practical advantages. It greatly simplifies the numerical complexity of the transfer process and makes it easier to specify regions that should move similarly.

The concept of deformation transfer can be posed as an analogy: given a pair of source meshes, $S$ and $S'$, and a target mesh $T$, generate a new mesh $T'$ such that the relationship between $T$ and $T'$ is analogous to the relationship between $S$ and $S'$. This form of reasoning was used to transfer drawing styles between two curves [Hertzmann et al. 2002]. Deformation transfer applies in the specific case where the relationship between $S$ and $S'$ is a continuous global deformation of the space and not an arbitrary relationship. This specialization enables optimal reproduction of the source deformation on the target mesh. Furthermore, deformation transfer does not require a common parameterization of the source and target meshes. Instead, it employs source and target meshes with matching reference poses much like facial animation uses a neutral face or skeleton-based techniques use a mesh in the T-pose.

When the deformation of a triangle mesh is purely skeleton-driven, transfer is more straightforward. A simple technique known as single-weight enveloping or skeleton-subspace deformation deforms mesh vertices by blending deformations of nearby skeleton bones. New target meshes can be swapped in by binding their vertices to the appropriate bones and setting the desired vertex weights. Furthermore, the motion of the skeleton can be parameterized by joint angles to define a set of natural control parameters for direct animation of the mesh with keyframing, or for retargeting motion from a skeleton of a different size and proportion [Gleicher 1998]. Deformation transfer is the first step toward the development of similar techniques for the animation of meshes with non-skeletal deformations or without an obvious skeletal structure. It generalizes the binding concept, which maps the motion of mesh vertices to the motion of a skeleton, by mapping deformations of the target mesh to deformations of the source.

Deformation transfer can have an immediate impact on example-based techniques, which currently rely on the artist to specify example shapes. Pose-space deformation, for example, corrects the "collapsing elbow" and other problems associated with simple skeleton-driven deformations by enabling the artist to sculpt corrective deformations [Lewis et al. 2000]. Once these corrections have been sculpted, deformation transfer can reduce the effort required to adapt them to new meshes. Bregler et al.'s cartoon-capture technique encodes a motion in the coefficients of a linear combination of meshes, which describe the animated character in a selection of key poses [2002]. Mapping these motions onto a different character requires an artist to recreate the new character in every single pose. Deformation transfer requires only one pose for the new character and automates the reproduction of the remaining poses. If the entire configuration space of the character is described in this manner [Ngo et al. 2000; Sloan et al. 2001], deformation transfer could lead to a technique for mapping the articulation from one character to another.

## 3 Deformation Transfer

The goal of deformation transfer is to transfer the change in shape exhibited by the source deformation onto the target. We represent the source deformation as a collection of affine transformations tabulated for each triangle of the source mesh. We use this representation because the non-translational portion of each affine transformation encodes the change in orientation, scale, and skew induced by the deformation on the triangle. However, the three vertices of a triangle before and after deformation do not fully determine the affine transformation since they do not establish how the space perpendicular to the triangle deforms. To resolve this issue, we add a fourth vertex in the direction perpendicular to the triangle. Let $\mathbf{v}_i$ and $\tilde{\mathbf{v}}_i$, $i \in 1\ldots3$, be the undeformed and deformed vertices of the triangle, respectively. We compute a fourth undeformed vertex as

$$\mathbf{v}_4 = \mathbf{v}_1 + (\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1) / \sqrt{|(\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)|} \quad (1)$$

and perform an analogous computation for $\tilde{\mathbf{v}}_4$. We scale the cross-product by the reciprocal of the square root of its length since this causes the perpendicular direction to scale proportional to the length of the triangle edges.

An affine transformation defined by the $3 \times 3$ matrix $\mathbf{Q}$ and displacement vector $\mathbf{d}$, which, for notational convenience, we write as $\mathbf{Q} + \mathbf{d}$, transforms these four vertices as follows:

$$\mathbf{Q}\mathbf{v}_i + \mathbf{d} = \tilde{\mathbf{v}}_i, \quad i \in 1\ldots4. \quad (2)$$

If we subtract the first equation from the others to eliminate $\mathbf{d}$ and rewrite them in matrix form treating the vectors as columns, we get $\mathbf{QV} = \tilde{\mathbf{V}}$ where

$$\begin{aligned} \mathbf{V} &= [\mathbf{v}_2 - \mathbf{v}_1 \;\; \mathbf{v}_3 - \mathbf{v}_1 \;\; \mathbf{v}_4 - \mathbf{v}_1] \\ \tilde{\mathbf{V}} &= [\tilde{\mathbf{v}}_2 - \tilde{\mathbf{v}}_1 \;\; \tilde{\mathbf{v}}_3 - \tilde{\mathbf{v}}_1 \;\; \tilde{\mathbf{v}}_4 - \tilde{\mathbf{v}}_1] \end{aligned} . \quad (3)$$

A closed form expression for $\mathbf{Q}$ is given by

$$\mathbf{Q} = \tilde{\mathbf{V}}\mathbf{V}^{-1}. \quad (4)$$

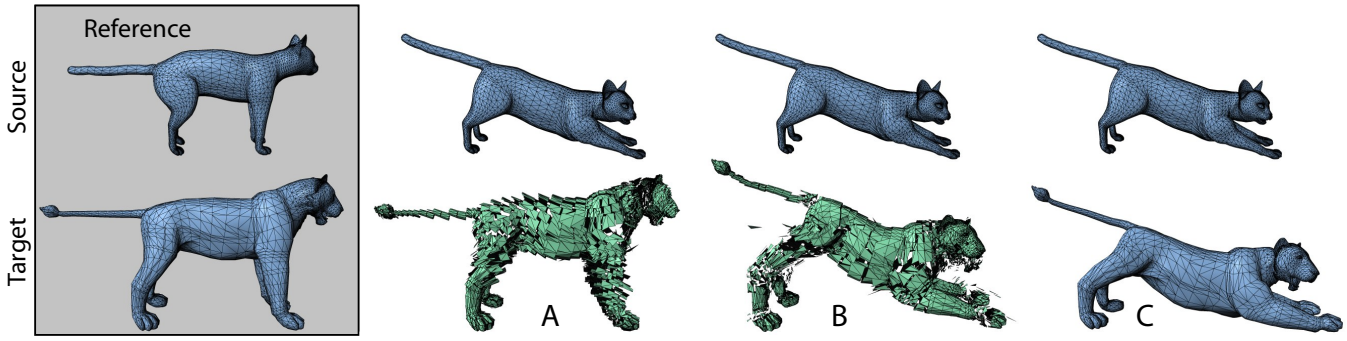


Figure 2: We encode a source deformation with an affine transformation for each source triangle and relate the transformations to the target through a user supplied triangle correspondence. (A) Using only the non-translational component of the source transformations transfers the change in orientation and scale to the target triangles but does not position them appropriately relative to their neighbors. (B) Using the source displacements gives a disconnected shape since consistency requirements are not enforced. (C) Deformation transfer solves a constrained optimization problem for a new set of target transformations that are as close as possible to the source transformations while enforcing the consistency requirements: shared vertices must be transformed to the same place.

We use Equation 4 to compute the source transformations $\mathbf{S}_1,\ldots,\mathbf{S}_{|S|}$ that encode the change in shape induced by the deformation, where $S$ refers to the set of triangle indices for the source mesh.

In order to relate the source deformation to the target mesh with the set of triangle indices $T$, the user supplies a mapping $M$ between the set indices for the source and target triangles:

$$M = \{(s_1,t_1),(s_2,t_2),\ldots,(s_{|M|},t_{|M|})\}. \tag{5}$$

A pair $(s_i,t_i)$ indicates that the target triangle with index $t_i$ should deform like the source triangle with index $s_i$. This mapping allows transferred deformations to originate from any region of the source mesh. There are no restrictions on $M$. In most cases, it is a general many-to-many mapping, but it can also be bijective (one-to-one and onto), surjective (onto), or not-onto. This generality enables transfer between meshes of different tessellations.

Our strategy is to transfer the source transformations via the correspondence map to the target. The non-translational portion $\mathbf{S}$ of a source affine transformation $\mathbf{S}+\mathbf{d}$ encodes the change in triangle shape induced by the source deformation. However, we cannot apply $\mathbf{S}$ directly to the corresponding target triangle since $\mathbf{S}$ encodes only the change in orientation and size and not the positioning of the triangle relative to its neighbors (Figure 2 A). Furthermore, we cannot use the source displacement vectors to resolve this problem since their lengths depend on the size and position of the source shape. Doing so yields a discontinuous shape (Figure 2 B) and exposes the fact that our deformation representation affords too many degrees of freedom. It allows the triangles to be transformed arbitrarily even though neighboring triangles share vertices.

In order to ensure that the affine transformations, when applied to the target mesh, are consistent with respect to one another, we require that shared vertices be transformed to the same place (Figure 3). For the set of target affine transformations $\mathbf{T}_1+\mathbf{d}_1\ldots\mathbf{T}_{|T|}+\mathbf{d}_{|T|}$ this requirement is

$$\mathbf{T}_j\mathbf{v}_i+\mathbf{d}_j = \mathbf{T}_k\mathbf{v}_i+\mathbf{d}_k, \quad \forall i,\ \forall j,k\in p(v_i), \tag{6}$$

where $p(\mathbf{v}_i)$ is the set of all triangles that share vertex $\mathbf{v}_i$.

In order to transfer the source deformation onto the target mesh while maintaining these consistency requirements (Figure 2 C), deformation transfer minimizes the difference between the non-translational components of the source and target transformations and enforces the consistency constraints in Equation 6 by solving the following constrained optimization problem for the target

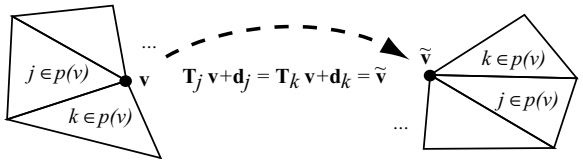


Figure 3: In order to maintain consistency, the affine transformations for all triangles $j,k\in p(\mathbf{v})$ that share vertex $\mathbf{v}$ must transform $\mathbf{v}$ to the same position.

affine transformations:

$$\min_{\mathbf{T}_1+\mathbf{d}_1\ldots\mathbf{T}_{|T|}+\mathbf{d}_{|T|}} \sum_{j=1}^{|M|} \left\|\mathbf{S}_{s_j}-\mathbf{T}_{t_j}\right\|_F^2 \tag{7}$$
$$\text{subject to} \quad \mathbf{T}_j\mathbf{v}_i+\mathbf{d}_j = \mathbf{T}_k\mathbf{v}_i+\mathbf{d}_k, \quad \forall i,\ \forall j,k\in p(v_i).$$

The matrix norm $||\cdot||_F$ is the Frobenius norm, or the square root of the sum of the squared matrix elements.

A solution of this optimization problem defines a continuous deformation of the target mesh up to a global translation. The global translation can be defined explicitly by setting the displacement $\mathbf{d}_i$ for any target triangle. In addition, other positional constraints such as foot placement can also be added.

## 4 Vertex Formulation

Although the formulation of deformation transfer in Equation 7 can be solved with quadratic programming techniques, a more efficient method eliminates the constraints by reformulating the problem in terms of vertex positions. The key idea is to define the unknown target transformations in terms of the triangles' vertices. Then, rather than solving for the entries of the affine transformations, we solve directly for the deformed vertex positions. This method satisfies all constraints because, by construction, any shared vertex will be transformed to the same location.

For each target triangle, we add a fourth undeformed vertex (Equation 1) and write the non-translational portion of the affine transformation in terms of the undeformed and deformed vertices $\mathbf{T}=\tilde{\mathbf{V}}\mathbf{V}^{-1}$. The elements of $\mathbf{V}^{-1}$ depend on the known, undeformed vertices of the target shape. The elements of $\tilde{\mathbf{V}}$ are the coordinates of the unknown deformed vertices. Thus, the elements of $\mathbf{T}$ are linear combinations of the coordinates of the unknown deformed vertices.

Given this definition, we rewrite the minimization problem as

$$\min_{\tilde{\mathbf{v}}_1 \ldots \tilde{\mathbf{v}}_n} \quad \sum_{j=1}^{|M|} \left\| \mathbf{S}_{s_j} - \mathbf{T}_{t_j} \right\|_F^2 . \tag{8}$$

Since the target transformations are defined in terms of the unknown deformed target vertices, the minimization is over the vertices themselves and the continuity constraints are implicitly satisfied. Positional vertex constraints can be enforced by simply treating a vertex as a constant rather than as a free variable.

The solution to this optimization problem is the solution to a system of linear equations. Rewriting the problem in matrix form yields

$$\min_{\tilde{\mathbf{v}}_1 \ldots \tilde{\mathbf{v}}_n} \quad ||\mathbf{c} - \mathbf{A}\tilde{\mathbf{x}}||_2^2 \tag{9}$$

where $\tilde{\mathbf{x}}$ is a vector of the unknown deformed vertex locations, $\mathbf{c}$ is a vector containing entries from the source transformations, and $\mathbf{A}$ is a large, sparse matrix that relates $\tilde{\mathbf{x}}$ to $\mathbf{c}$. Setting the gradient of the objective function to zero gives the familiar normal equations:

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}\tilde{\mathbf{x}} = \mathbf{A}^{\mathrm{T}}\mathbf{c} \tag{10}$$

The entries in $\mathbf{A}$ depend only on the target mesh's undeformed vertex locations. Furthermore, the system is separable in the spatial dimension of the vertices. Thus, for each source/target pair, we compute and store the LU factorization of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ only once. Retargeting any source deformation onto the target mesh only requires performing backsubstitution to solve separately for the $x$, $y$, and $z$ components of the deformed target vertices. For efficiency, we use a sparse LU solver [Davis 2003]. Since the columns of $\mathbf{A}$ correspond to the unknown deformed target vertices, and since we add an extra vertex for each triangle, the number of columns of $\mathbf{A}$ (and hence the number rows and columns of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$) is equal to the number of vertices plus the number of triangles of the target mesh. Table 1 lists the vertex and triangle counts for the meshes in this paper, and Table 2 lists the factorization and average backsubstitution times.

## 5 Correspondence

The correspondence between the source and target triangles defines how the deformation of the source mesh should be transferred to the target. We aid the creation of this mapping with a tool that automatically computes the triangle correspondence from a small set of $m$ user selected marker points. Our correspondence technique is an iterated closest point algorithm with regularization, aided by user selected marker points, that deforms the source mesh into the target mesh. Then, it computes the triangle correspondence by searching for pairs of source and target triangles whose centroids are in close proximity. Figure 4 illustrates this process. Our correspondence system is similar to the template fitting procedure described by Allen, Curless, and Popović [2003] but developed in the context of our numerical framework.

The correspondence system solves a minimization problem similar to the one we use for deformation transfer, but the objective function is designed to deform one mesh *into* the other, rather than deforming it *like* the other deforms. The user controls the deformation by supplying a set of marker points specified as pairs of source and target vertex indices. Each pair indicates that the source vertex, after deformation, should match the location of the target vertex. These markers are enforced as constraints in the minimization. The objective function contains a term that enforces deformation smoothness, one that prevents over smoothing, and one that moves the source vertices to the target mesh.

Deformation smoothness, $E_S$, indicates that the transformations for adjacent triangles should be equal. For a mesh with $n$ vertices, we let $T$ be the set of triangle indices and $\mathbf{T}_1 + \mathbf{d}_1 \ldots \mathbf{T}_{|T|} + \mathbf{d}_{|T|}$ be the affine transformations that define the deformation. Then,

$$E_S(\mathbf{v}_1 \ldots \mathbf{v}_n) = \sum_{i=1}^{|T|} \sum_{j \in \mathrm{adj}(i)} ||\mathbf{T}_i - \mathbf{T}_j||_F^2 . \tag{11}$$

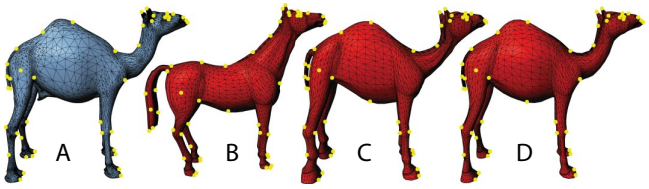


Figure 4: The correspondence algorithm deforms the source mesh into the target, controlled by user selected marker points shown in yellow. (A) Target mesh. (B) Source mesh. (C) Source mesh after the first phase of deformation where the closest valid point term is ignored. (D) Final deformed mesh using all three terms of the objective function.

Here, $\mathbf{T}_1, \ldots, \mathbf{T}_{|T|}$ are defined in terms of the target vertices according to Equation 4, and $\mathrm{adj}(i)$ is the set of triangles adjacent to triangle $i$. Note that this term is minimized when the *change* in deformation, and not the mesh itself, is smooth. For example, regardless of the smoothness of the mesh, any rigid transformation applied to all triangles is a valid minimum for $E_S$.

Deformation identity, $E_I$, is minimized when all transformations are equal to the identity matrix:

$$E_I(\mathbf{v}_1 \ldots \mathbf{v}_n) = \sum_{i=1}^{|T|} ||\mathbf{T}_i - \mathbf{I}||_F^2 . \tag{12}$$

The purpose of this term is to prevent the deformation smoothness term from generating a drastic change in the shape of the mesh in order to achieve optimal smoothness.

The closest valid point term, $E_C$, indicates that the position of each vertex of the source mesh should be equal to the closest valid point on the target mesh.

$$E_C(\mathbf{v}_1 \ldots \mathbf{v}_n, \mathbf{c}_1 \ldots \mathbf{c}_n) = \sum_{i=1}^{n} ||\mathbf{v}_i - \mathbf{c}_i||^2 , \tag{13}$$

where $\mathbf{c}_i$ is the closest valid point on the source mesh to target vertex $i$. When computing the closest valid point, vertex normals of the source mesh are compared with triangle normals of the target mesh and a difference in orientation of less than $90^\circ$ indicates a valid point. A grid-based spatial sorting algorithm accelerates the closest point computation.

To compute the deformed vertices $\tilde{\mathbf{v}}_1 \ldots \tilde{\mathbf{v}}_n$ of the source mesh, we define the following minimization problem

$$\begin{aligned} \min_{\tilde{\mathbf{v}}_1 \ldots \tilde{\mathbf{v}}_n} \quad & E(\mathbf{v}_1 \ldots \mathbf{v}_n, \mathbf{c}_1 \ldots \mathbf{c}_n) = w_S E_S + w_I E_I + w_C E_C \\ & \text{subject to} \quad \tilde{\mathbf{v}}_{s_k} = \mathbf{m}_k, \quad k \in 1 \ldots m \end{aligned} \tag{14}$$

where $w_S$, $w_I$, and $w_C$ are weights, $s_k$ is the source vertex index for marker $k$, and $\mathbf{m}_k$ is the position of marker $k$ on the target mesh. We solve this minimization in two phases. In the first phase, we ignore the closet point term by using weights $w_S = 1.0$, $w_I = 0.001$, and $w_C = 0$. We solve the problem for the deformed source mesh (Figure 4 B). The marker points of the deformed mesh will match exactly since they are specified as constraints, and the rest of the mesh will be carried along by the smoothness and identity terms. We use this initial estimation to compute a set of valid closest points. Then, in the second phase, we solve the same problem increasing $w_C$ each time and updating the closest points after each iteration. Preserving $w_S = 1.0$ and $w_I = 0.001$ while increasing $w_C$ in four steps from 1.0 to 5000.0 produced good results in our tests. Each time the minimization problem is solved, the source mesh is deformed from its original undeformed state. Since $w_C$ increases, the source mesh more closely approximates the target mesh after each iteration (Figure 4 C).

Once the source mesh has been deformed to match the shape of the target mesh, we compute the triangle correspondences by comparing the centroids of the deformed source and target triangles.

| Mesh | Vertices | Triangles |
|---|---|---|
| Horse | 8,431 | 16,843 |
| Camel | 21,887 | 43,814 |
| Cat | 7,200 | 14,410 |
| Lion | 5,000 | 9,996 |
| Face | 29,299 | 58,836 |
| Head | 15,941 | 31,620 |
| Flamingo | 26,907 | 52,895 |

Table 1: Number of vertices and triangles for our example meshes.

| Example | Number of Markers | LU Factorization | Back-substitution |
|---|---|---|---|
| Horse/Camel | 65 | 1.559s | 0.293s |
| Cat/Lion | 77 | 0.299s | 0.057s |
| Face/Head | 42 | 1.252s | 0.298s |
| Horse/Flamingo | 73 | 1.495s | 0.406s |

Table 2: The number of correspondence markers used and the timing results for our examples on a 3.0GHz Pentium IV machine.

Two triangles are compatible if their centroids are within a certain threshold of each other and the angle between their normals is less than 90°. This compatibility test prevents two nearby triangles with disparate orientation (e.g., triangles from the upper and lower lips of a face) from entering the correspondence. For each triangle of the deformed source, we compute the closest compatible triangle (if any) of the target mesh and add the pair to the correspondence list. Likewise, for each triangle of the target, we compute the closest compatible triangle of the deformed source mesh and add that pair. This process ensures that all triangles of the source and target meshes, subject to the compatibility restriction, will be listed among the correspondences. A target triangle may correspond to many source triangles, and vice versa. This feature allows our deformation transfer method to accommodate meshes with differing numbers of vertices and triangles.

## 6 Results

Figure 1 shows deformations of a horse transfered onto a camel. The reference horse mesh, shown in the gray box, is deformed into seven key poses. The key poses include obvious skeletal deformations such as bending of the legs or neck as well as more subtle skin deformations like stretching near the joints. The input to our algorithm is the reference horse mesh, the seven deformed horse poses, the reference camel mesh, and the correspondence between the two reference meshes. Given this data, deformation transfer generates seven new camel poses by transferring the source deformations onto the reference camel. Both the gross skeletal changes as well as the more subtle skin deformations are faithfully reproduced on the camel. Figure 5 demonstrates a similar set of deformations. Here, key poses of a cat are retargeted onto a lion. Since deformation transfer copies the *change* in shape induced by the deformation, we require the source and target reference meshes to have the same kinematic pose when skeletal deformations are retargeted.

While the deformations of Figures 1 and 5 are primarily skeletal in nature, Figure 6 demonstrates the effectiveness of our approach with non-rigid deformations. Here, the horse collapses as if it were made of a rubber sheet. Its legs buckle and its entire body falls to the ground, folding on top of itself. The deformations are transfered to the camel, and its body buckles and collapses similarly.

In the accompanying video, we use deformation transfer to retarget two deformations that vary continuously through time. First, we transfer a gallop gait from the horse to the camel, and then we transfer the collapsing motion from Figure 6. In order to resolve the global positioning of the camel over time and to enforce foot/ground contact, we extracted the positions of one vertex on each foot of the horse over time, performed an overall scaling to better match the larger size of the camel, and added vertex constraints to match a vertex on each camel foot to these positions. Deformation transfer then copies the horse deformation onto the camel while simultaneously satisfying the vertex constraints. Because computing the source deformations as well as mapping them to the target are both linear operations, temporal consistency of the source animation and vertex constraints results in a temporally coherent target animation.

In Figure 7, facial expressions of a real person, acquired with a 3D scanning system, are transfered onto a digital character. A great deal of expressiveness—especially around the eyes and nose—is captured and adapted to the target head. This type of transfer might be used when a digital stand-in must replace a real actor, or to map the facial expressions of a voice actor onto an animated character.

Since the scanned data is in the form of face masks and the target mesh consists of an entire head and neck, the mapping between the source and target triangles is not onto. Only a subset of the target triangles (those of the front of the face) are listed in the correspondence. We specify that the deformation of the remaining target triangles should be minimal by mapping them to the identity matrix.

Our deformation transfer technique was designed for the case where there is a clear semantic correspondence between the source and target. We chose anatomically similar meshes to demonstrate our results since they have an obvious mapping (i.e., leg to leg, head to head, etc.). In Figure 8, we challenge this assumption by transferring the horse deformation onto a flamingo mesh. The correspondence is ambiguous as the flamingo has only two legs, no tail, and a beak. We mapped the flamingo's legs to the horse's front two legs, the flamingo's body to the entire horse's body including its tail, and its beak to the horse's head. Building this mapping pushes the limits of our correspondence system as the flamingo's "S"-shaped neck must be unbent to match the horse's straight neck. The deformation smoothness term (Equation 11) fights against these large local deformations, requiring the user to select many marker points along the neck. However, once the correspondence has been adequately specified, the flamingo faithfully deforms like the horse. Of course, a real flamingo's hips bend in the opposite direction of a horse's front hips, which demonstrates a reason why deformation transfer between anatomically different meshes may not be appropriate.

None of the source and target meshes in our examples share the same number of vertices, triangles, or connectivity. Table 1 lists this geometric information about each model, and Table 2 gives timing results for each example. Our method is extremely fast. For example, the camel mesh, consisting of $21,887$ vertices and $43,814$ triangles, required only 1.559 seconds for factorization and 0.293 seconds on average to solve for each retargeted pose. The user time required to add the markers and compute the correspondence for each example was under one hour.

## 7 Conclusion

Deformation transfer is a general mesh based technique that transfers deformations exhibited by a source mesh onto a different target mesh. The technique transfers between meshes with different mesh structure (number of vertices, number of triangles, and connectivity). The process is numerically efficient. A precomputation step factors and stores the system of equations for a source/target pair. Transferring a new deformation involves a backsubstitution with the factored system.

The user controls the transfer process by supplying a mapping between the triangles of the source and the triangles of the target that identifies which parts should move similarly. Our correspondence tool assists the user by automatically computing the triangle correspondence from a small number of user supplied markers.

In order to perform deformation transfer, the source and target deformations are represented as affine transformations. The known source deformations are mapped to the target via the correspondence map. We solve a constrained optimization for the target deformation that matches the source transformations as closely as pos-

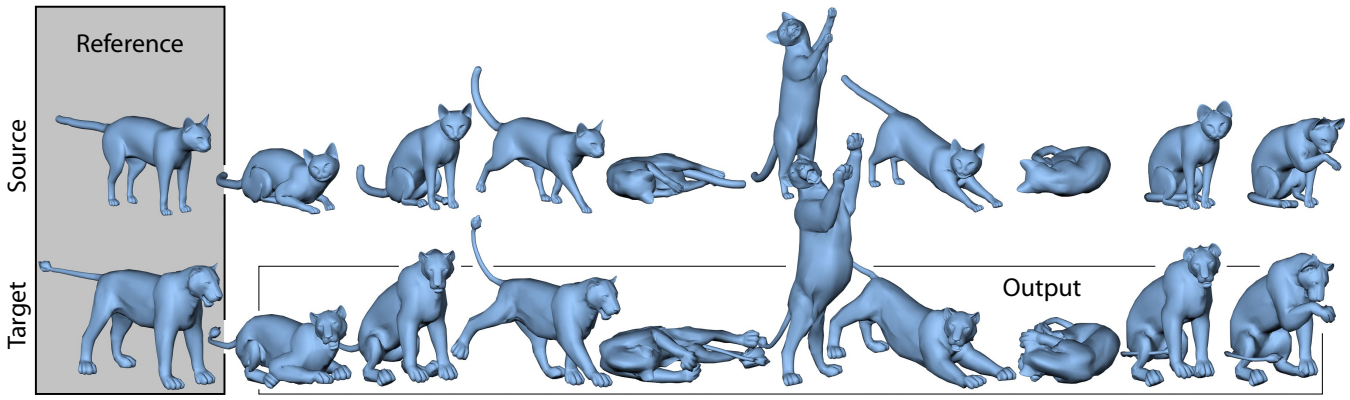


Figure 5: Cat poses retargeted onto a lion.

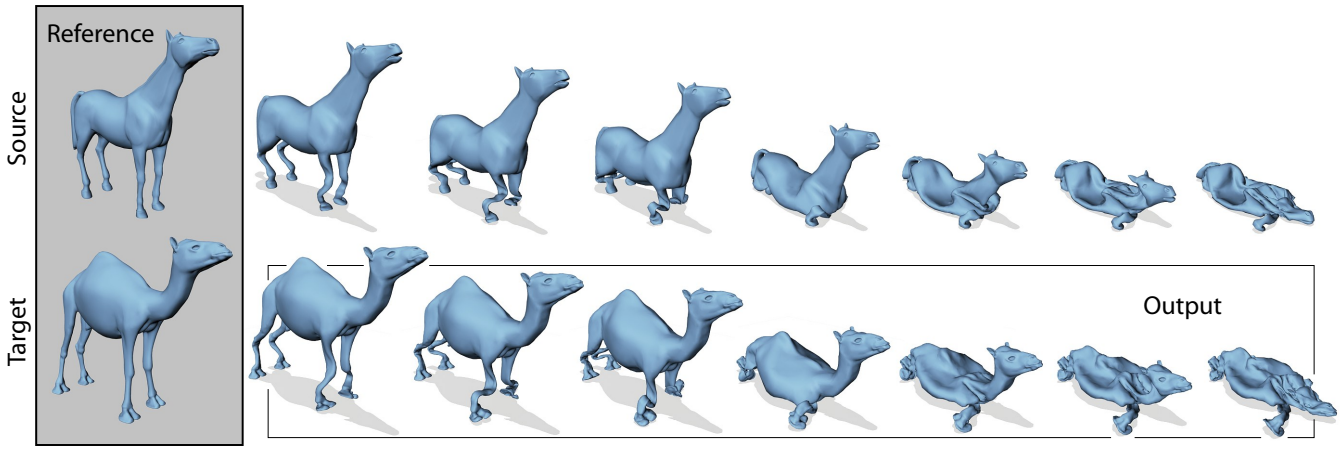


Figure 6: The horse deformation, collapsing as if it were made of a rubber sheet, is transferred to the camel.

sible while maintaining consistency constraints. For efficiency, we use a vertex formulation of this problem that satisfies the constraints implicitly.

Several limitations of our current system point to directions of future work. Our method allows limited control over the deformation transfer process in the form of the correspondence map and vertex constraints, but provides no direct way to fine-tune the solution aside from changing the source deformation and reapplying deformation transfer. In particular, it cannot transfer the animation controls used to generate the deformation in the first place. Ideally, deformation transfer would carry over the controls as well, allowing the retargeted deformation to be fine-tuned for the target shape. A general technique to adapt animation control knobs from one character to another is an open problem in computer graphics.

The optimization problem that we solve is unique up to a global translation. Thus, the global position must always be specified in some way by the user. When retargeting only key poses, as in Figures 1 and 5, it is easy to resolve the global position by fixing one vertex in place. However, when retargeting an entire animation sequence, the position must be specified at each point in time. A positional constraint specified only during selected intervals (such as during ground contact) will result in "popping" artifacts when the constraint becomes active. In the accompanying video, we resolved the issue by constraining the camel's feet to match the horse's feet at each point in time. But, if two meshes have very different proportions, it may be more difficult to formulate an appropriate constraint.

One way to approach this problem is to directly address the temporal dimension in the retargeting process in order to enforce temporal coherence as well as reduce the global positioning problem to the specification of positional constraints only during key events such as foot/ground contact. However, a formulation of deformation transfer over time would increase the numerical complexity of the optimization, requiring a different numerical approach. With our direct LU solver, we have successfully transfered deformations onto a target mesh with 400k triangles. However, at this point the LU solver approaches memory limitations. The LU factorization, with a considerable amount of swapping, took 95 seconds, and backsubstitution took 6.5 seconds. For extremely large meshes, a multigrid solver would likely outperform our direct method.

Perhaps the most conspicuous limitation of our technique is the requirement of gross similarity between the source and target meshes. Transferring deformation between drastically different meshes is an open problem in computer graphics that presents two primary challenges. First, it requires a more versatile technique to relate the source and target to one another that can accommodate ambiguous and arbitrary mappings. Second, it requires a method to appropriately *adapt* the deformation to the target, rather than simply transferring it directly without modification.

## Acknowledgments

The authors would like to thank Daniel Vlasic for his assistance on an earlier version of this project, Ray Jones for his insightful comments, Charles Han and Andrew Elliott for their help in preparing the figures, and Shuang You for an earlier version of the correspondence system. This research was sponsored by the Deshpande Center for Technological Innovation.


Figure 7: Scanned facial expressions cloned onto a digital character.

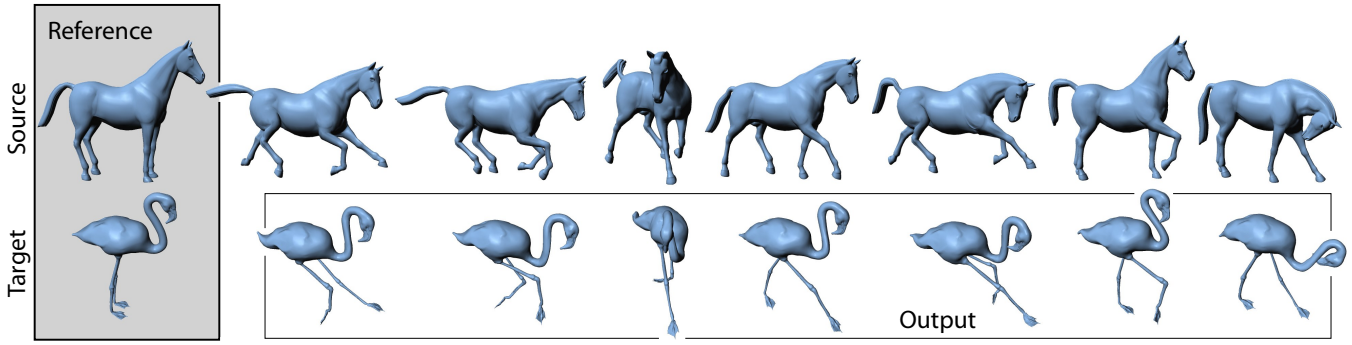


Figure 8: Horse poses mapped onto a flamingo.

## References

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-rigid-as-possible shape interpolation. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, 157–164.

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2003. The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Transactions on Graphics 22*, 3 (July), 587–594.

BARR, A. H. 1984. Global and local deformations of solid primitives. In *Computer Graphics (Proceedings of ACM SIGGRAPH 84)*, vol. 18, 21–30.

BREGLER, C., LOEB, L., CHUANG, E., AND DESHPANDE, H. 2002. Turning to the masters: Motion capturing cartoons. *ACM Transactions on Graphics 21*, 3 (July), 399–407.

DAVIS, T. A. 2003. Umfpack version 4.1 user guide. Tech. rep., University of Florida. TR-03-008.

GLEICHER, M. 1998. Retargeting motion to new characters. In *Proceedings of ACM SIGGRAPH 1998*, Computer Graphics Proceedings, Annual Conference Series, 33–42.

HERTZMANN, A., OLIVER, N., CURLESS, B., AND SEITZ, S. M. 2002. Curve analogies. In *Eurographics Workshop on Rendering 2002*, 233–246.

HSU, W. M., HUGHES, J. F., AND KAUFMAN, H. 1992. Direct manipulation of free-form deformations. In *Computer Graphics (Proceedings of ACM SIGGRAPH 92)*, vol. 26, 177–184.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformations: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, 165–172.

LITWINOWICZ, P. C. 1991. Inkwell: A 2 1/2-d animation system. In *Computer Graphics (Proceedings of ACM SIGGRAPH 91)*, vol. 25, 113–122.

NGO, T., CUTRELL, D., DANA, J., DONALD, B., LOEB, L., AND ZHU, S. 2000. Accessible animation and customizable graphics via simplicial configuration modeling. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, 403–410.

NOH, J., AND NEUMANN, U. 2001. Expression cloning. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 277–288.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *Computer Graphics (Proceedings of ACM SIGGRAPH 86)*, vol. 20, 151–160.

SINGH, K., AND FIUME, E. L. 1998. Wires: A geometric deformation technique. In *Proceedings of ACM SIGGRAPH 1998*, Computer Graphics Proceedings, Annual Conference Series, 405–414.

SLOAN, P.-P. J., ROSE, III, C. F., AND COHEN, M. F. 2001. Shape by example. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, ACM Press, 135–143.