DOI: 10.1111/cgf.12507

COMPUTER GRAPHICS *forum*
Volume 34 (2015), number 1 pp. 216–227

Example-Based Retargeting of Human Motion to Arbitrary Mesh Models

Ufuk Celikcan[1], Ilker O. Yaz[2] and Tolga Capin[3,*]

[1]Department of Computer Engineering, Hacettepe University, Ankara, Turkey
celikcan@acm.org
[2]Microsoft, Redmond, WA, USA
ilkeryaz@gmail.com
[3]Department of Computer Engineering, TED University, Ankara, Turkey
tolga.capin@gmail.com

Abstract

We present a novel method for retargeting human motion to arbitrary 3D mesh models with as little user interaction as possible. Traditional motion-retargeting systems try to preserve the original motion, while satisfying several motion constraints. Our method uses a few pose-to-pose examples provided by the user to extract the desired semantics behind the retargeting process while not limiting the transfer to being only literal. Thus, mesh models with different structures and/or motion semantics from humanoid skeletons become possible targets. Also considering the fact that most publicly available mesh models lack additional structure (e.g. skeleton), our method dispenses with the need for such a structure by means of a built-in surface-based deformation system. As deformation for animation purposes may require non-rigid behaviour, we augment existing rigid deformation approaches to provide volume-preserving and squash-and-stretch deformations. We demonstrate our approach on well-known mesh models along with several publicly available motion-capture sequences.

Keywords: animation systems, deformations, motion capture, retargeting

ACM CCS: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism — Animation

1. Introduction

Recent advances in modelling and deformation, in parallel with motion-capture technologies, have made creating and animating virtual humans a common achievement. Animating virtual human meshes can be accomplished in many ways, including through rigging, keyframing and inbetweening deformable models or by physics-based simulations. However, these processes require prior training about the animation technique, and are tedious to use for non-experts.

As demonstrated by the increasing popularity of player-generated art assets in video games, there is a growing need for animation tools designed for novice users. For example, it is desirable to load an arbitrary 3D mesh model and animate it directly with a simple user interface. Our work addresses this problem with a novel method for generating animations of arbitrary 3D mesh models from human motion-capture data.

While skeletal deformation techniques remain the most common way to animate mesh models, these require extra structural definitions (e.g. a skeleton or cage) for the input mesh. Our goal is to relieve the user from building such a structure, and hence, our solution follows a surface-based deformation approach. In particular, we rely on the principles of as-rigid-as-possible mesh deformation [SA07], with the purpose of preserving the shape of the mesh while satisfying given deformation constraints. We extend this deformation framework to also consider squash-and-stretch behaviour in animations by relieving the rigidity constraints to a certain extent. Thus, we aim for achieving deformation results that can be categorized between rigid and cartoon-like.

Another key feature of our solution is that it provides an example-based retargeting approach, which is capable of animating non-humanoid mesh models that are topologically very different from human models. In our approach, using a simple interface, the user needs only to provide a few correspondent poses of the humanoid skeleton and the mesh model. These poses are then used to construct a mapping function between the source skeleton and the target mesh by our spacetime solver, which is also capable of handling kinematic and temporal constrains in the retargeting step.

*The authors developed this work while at the Department of Computer Engineering, Bilkent University, Turkey.

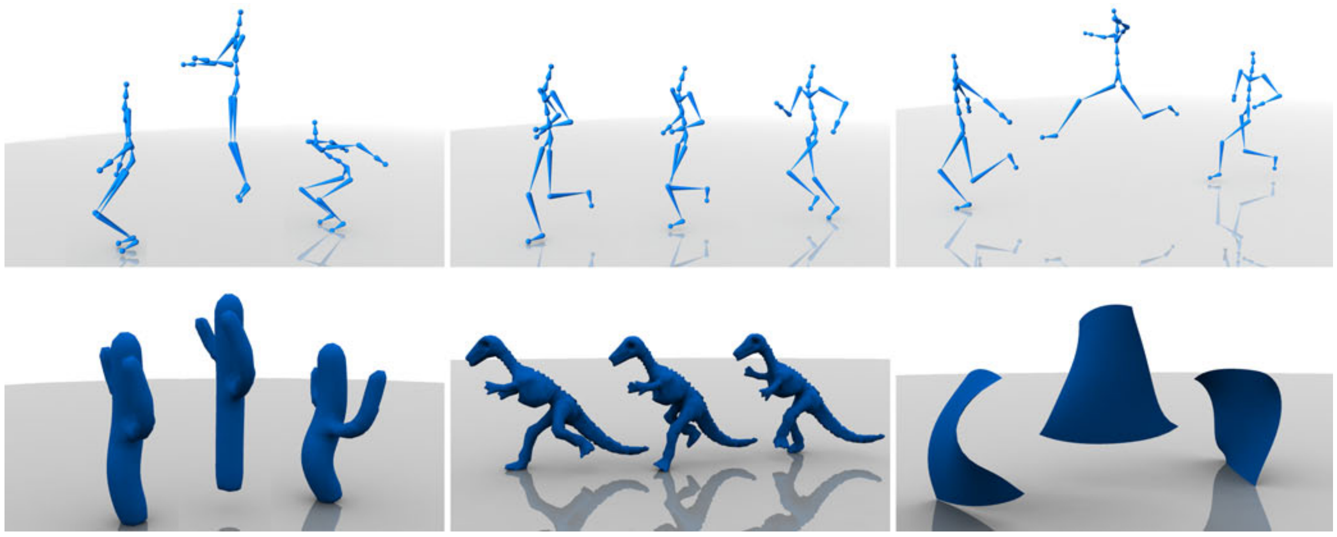


Figure 1: *Retargeting human motions to arbitrary mesh models.*

The contributions of our work include: (1) an example-based trajectory retargeting method for animating non-humanoid meshes; (2) an augmented as-rigid-as-possible deformation solution that is also capable of generating desired effects in animations such as squash and stretch and (3) a simple interface for retargeting human motion to arbitrary mesh models.

2. Related Work

Our work is related to the motion-retargeting problem in computer animation, and to the surface-based mesh editing in geometry processing. As motion retargeting and mesh editing have a rich literature, we survey only the most related work in both fields.

Motion Retargeting. One of the earliest approaches for motion retargeting, proposed by Gleicher [Gle98], uses a skeleton for rigging the mesh model similar to that of the skeleton for the motion data, but with different bone proportions and joint degrees of freedom. The motion from the source skeleton can then be retargeted to the target skeleton by minimizing the change in the motion while preserving the motion constraints. This approach is only feasible if the structure of the mesh is similar to the skeleton (e.g. a humanoid-like mesh with no topological differences). To overcome this restriction, several solutions are proposed. For example, Monzani *et al.* [sMBBT00] employ an intermediate skeleton while mapping source and target skeletons with different hierarchies using an integrated inverse kinematic engine for satisfying the motion constraints.

More recent approaches, such as Ikemoto *et al.*'s [IAF09] and Yamane *et al.*'s work [YAH10], construct statistical mapping between the source and target skeleton, by using a set of training data (a sequence of poses for the former and pose-to-pose correspondence for the latter) from different statistical models. Yamane *et al.* also address the issues of satisfying contact point constraints, improving physical realism and achieving better global transformation. To assure the motions are morphologically independent, Hecker *et al.* [HRE*08] propose an online retargeting system that relies on metadata (e.g. semantic structure, constraints) defined from motions created by animators. These generic motions can then be retargeted to specific characters with the help of a specialized inverse kinematic system at runtime. Compared to the above-mentioned studies, Hecker *et al.*'s system allows the animator to directly describe metadata for motions instead of using training data to extract these metadata. Bharaj *et al.* [BTST12] retarget skeletal motions to multi-component characters. After creating an appropriate skeleton for a given character, they construct a mapping between the joints of the input and created skeletons. Their mapping algorithm is designed to handle different morphologies automatically, where neither training data nor metadata are necessary. However, eliminating training leads to a lack of semantic correspondence between the source and target skeletons and their poses, limiting the retargeting process to direct transfer only.

The mentioned methods are based on skeleton-to-skeleton retargeting and require a rigged mesh to produce the final results. For a simpler interface for retargeting of human motion to arbitrary mesh models, it might be desirable to avoid rigging and skinning. Furthermore, an input mesh model might not even have an obvious skeletal structure.

Mesh Deformation. While skeletal deformation remains the most common way to animate mesh models, surface- and space-based methods also exist for mesh deformation. Among the space-based methods, Joshi *et al.* [JMD*07] and Lipman *et al.* [LLCO08] use a cage-based representation for character animation and manipulation of mesh models. These approaches are applicable to generic mesh models, where mesh deformation is driven by a surrounding cage, with the deformed cage determining how the resulting mesh preserves the general shape and local details. For our problem, the main disadvantage of these methods is that a carefully constructed cage is a pre-requisite for high-quality deformations.

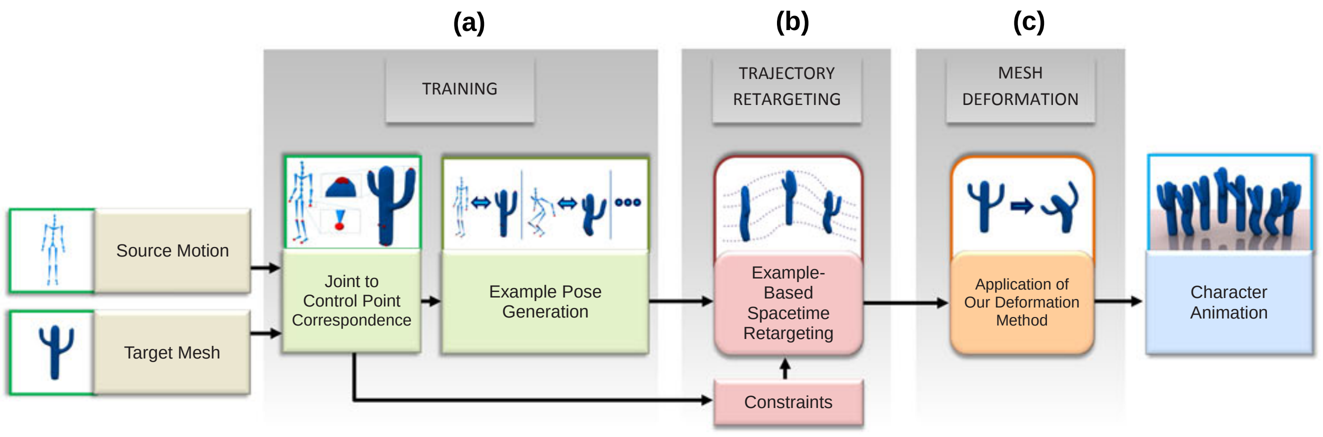


Figure 2: *Overview of our method. (a) Initially, the user designates significant joints and pairs these with the appropriate vertices of the target mesh by denoting them as mesh control points. The user then provides example pose-to-pose matches, selecting key poses from the source motion sequence and using our built-in shape deformation system to create corresponding mesh poses by simply dragging the control points. (b) For each frame in the input motion sequence, the corresponding positions of the control points are found by our example-based trajectory retargeting solution. This phase may further employ user- or system-specified constraints. (c) Using the calculated control point positions, the mesh deformation phase determines the deformation of the target mesh for each frame and produces the retargeted mesh animation.*

The main purpose of surface-based methods is to preserve the shape of the mesh while satisfying given deformation constraints. The shape of the mesh is represented by using variations of differential coordinate representations [LSCo*04, SCOL*04], where the aim is making the representation invariant to editing operations (e.g. translation, rotation) as much as possible. Surface-based deformations can also be used in animation. To provide surface detail and volume-preserving deformations, Zhou *et al.* [ZHS*05] propose a volumetric representation using differential coordinates of the mesh while applying deformation energy minimization. In their work, motions from 2D cartoons are retargeted using curves as control structures, which provide geometric correspondences between a mesh and a cartoon. Baran *et al.* [BVGP09] propose a method to retarget motion from one mesh to another by learning the semantic relationship between meshes from pose-to-pose correspondent training data. The key point in their work is developing a shape representation that is suitable for measuring mesh similarities, and for interpolating weighted meshes.

The overall animation transfer effect has been studied in previous mesh animation editing work [XZY*07]. Xu *et al.* provide a technique for editing of deforming mesh sequences given as input, and their work does not directly address the mapping of a skeletal animation to an arbitrary mesh. Their handle-based deformation mixing approach essentially applies a geometric mapping, whereas our method provides a semantic mapping, i.e. the mesh motion does not have to be spatially close to the skeleton. Furthermore, our deformation method is built upon and extends the as-rigid-as possible (ARAP) method for volume preservation and trajectory following, therefore we see that our method is a more convenient tool in interactive key frame editing.

Our work combines spacetime formulation of animation with shape-preserving deformation. This hybrid approach extends existing shape-preserving deformation methods to provide volume-preserving, less-rigid and more-cartoon-like deformations. Our method also improves the traditional spacetime retargeting approach by making it compatible with example-based retargeting.

3. Overview

Our method accepts as input a motion sequence (either keyframed or captured) and a 3D mesh. The input mesh is to be of a single shell, which is customary with 3D models. Our method consists of three phases (Figure 2). The first phase is the training phase, where the user first selects a desired number of joints from the source skeleton and the vertices of the target mesh with a simple interface. This phase allows the user to specify joints that are semantically important for the motion and/or appropriate for the target mesh structure. While there are methods for automatically rigging a mesh with a given skeleton [BP07], in which the corresponding parts of the source skeleton and the target mesh are identified by the system itself, these methods necessitate a skeleton structure, and further, require the structure of the skeleton to be compatible with the mesh.

In the next part of the training phase, the user provides pose-to-pose training data to construct the mapping function between the joints of the skeleton and the control points of the mesh. For this purpose, the user selects only a few key poses from the source motion sequence and uses our built-in shape deformation system to create corresponding mesh poses by simply dragging the selected control points. Selecting the example poses is essential because we pursue a semantical correspondence between the source skeleton and the target mesh rather than a direct geometrical mapping. Thus, our method conceptually falls in the same category as [YAH10, BVGP09] and [SZGP05], which also utilize example poses so the user can specify desired retargeting semantics.

After these simple manual steps have been performed, the automatic retargeting phase follows an example-based retargeting approach to reflect the relationship between the example poses. For each frame in the input motion sequence, the corresponding positions of the mesh control points are found using our example-based trajectory retargeting method. This phase may further contain user- or system-specified constraints (Section 5). Using the calculated control point positions, the final phase determines the deformation of the target mesh, and produces the final retargeted mesh animation (Section 4).

We use Cartesian space for representing the humanoid skeleton joint positions and the mesh model poses. To transform the skeleton into the body's local coordinate frame, we only need to cancel the yaw angle together with global translation. We use the root joint as the origin of the coordinate frame, although any other point can be used for this purpose because our retargeting method is translation invariant. For the mesh model, we use its centre as the origin.

4. Mesh Deformation

Our surface-based mesh deformation method builds on the general-purpose deformation approach, in that the main purpose is preserving the shape of the mesh while satisfying given deformation constraints. However, unlike traditional general-purpose deformation systems, our mesh deformation is designed for animation (Figure 3). We consider the following to be desirable characteristics of our deformation algorithm:

1. Mesh deformation should be interactive and allow direct manipulation of the desired deformation because the user deforms the mesh with the same interface while providing example poses.
2. The solution should be sparse: selecting only a few control points on the mesh should give acceptable deformation results.
3. The deformation should be as shape preserving as possible while allowing deviations from rigidity to achieve squash-and-stretch effects, a key principle of animation.
4. The deformation should be as volume preserving as possible, another key principle of animation.

Considering the first characteristic, among existing methods in the literature, it might appear more plausible to employ linear deformation methods, such as [SCOL*04, LSLCO05], instead of non-linear methods, such as [SK04, SA07] However, linear methods require a large number of control points and/or explicit rotational constraints to achieve acceptable deformation, which contradicts the second goal. Assuming that the target mesh is reasonably complex, non-linear methods are more appropriate for satisfying the first and second goals. In our approach, we build upon *ARAP surface modelling* [SA07]. The traditional ARAP method satisfies the first two characteristics, but it fails to support the latter two.

Laplacian Surface Deformation. Laplacian coordinates are a way of representing the mesh differentially:

$$L(\mathbf{v}_i) = \mathbf{v}_i - \sum_{\mathbf{v}_j \in N(\mathbf{v}_i)} w_{ij}\mathbf{v}_j, \tag{1}$$

where $\mathbf{v}_i$ is the current vertex; $L(\mathbf{v}_i)$ is the Laplacian coordinate of $\mathbf{v}_i$; $N(\mathbf{v}_i)$ is the set of one-ring neighbour vertices, i.e. the *cell*, of $\mathbf{v}_i$ and w_{ij} is a weight between $\mathbf{v}_i$ and neighbour $\mathbf{v}_j$. Cotangent weights are used to eliminate the effect of non-uniform discretization. Considering the mesh as a whole, it is possible to generate a linear system:

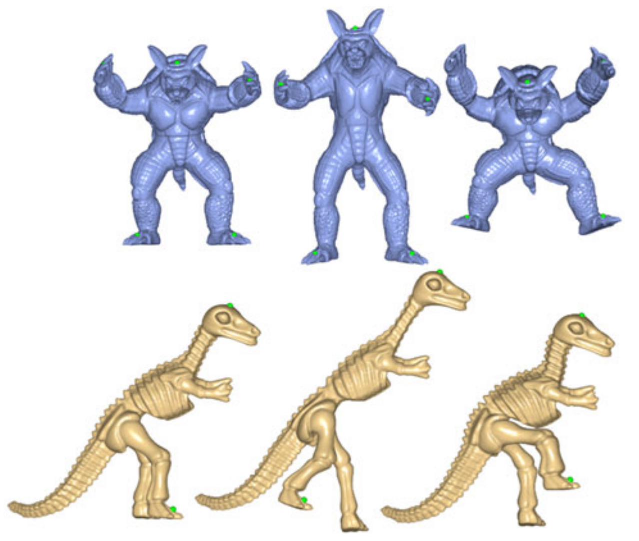


Figure 3: *Demonstrating our deformation solution on Armadillo and T-Rex models using a few control points to interactively deform meshes. The green markers denote the user-specified control points.*

$$\mathbf{LV} = \delta, \tag{2}$$

where $\mathbf{L}$ is an $n \times n$ square matrix, with each row filled according to Equation (1); $\mathbf{V}$ is an $n \times 1$ column vector consisting of global position of vertices and δ represents the Laplacian coordinates of the mesh. Integrating the user-selected control points to Equation (2) as soft constraints can be achieved by:

$$\begin{bmatrix} \mathbf{L} \\ \mathbf{I}_c \end{bmatrix} \mathbf{V}' = \begin{bmatrix} \delta \\ \mathbf{V}_c \end{bmatrix}, \tag{3}$$

where $\mathbf{I}_c$ is an $m \times n$ matrix for m control vertices, with each row containing a weight in the related column with the control vertex; and $\mathbf{V}_c$ contains the weighted positions of the control vertices. The overdetermined system in Equation (3) is solved in a least-squares sense to find $\mathbf{V}'$, which preserves the Laplacian coordinates of the mesh (δ) and the positions of the control points ($\mathbf{V}_c$) as much as possible. Soft constraints can be converted into hard constraints by transferring the related vertex from the left side to the right in Equation (1) [SA07].

Squash-and-Stretch Deformation. The main characteristic of the traditional ARAP deformation is finding optimum rotations that preserve edge lengths under given constraints. However, while animating mesh models, it is more desirable to have a bias towards a pose that conveys emotion, rather than towards a pose where local details of the mesh are preserved. In other words, acceptability and cartoon-like features of the mesh's global shape suppress the

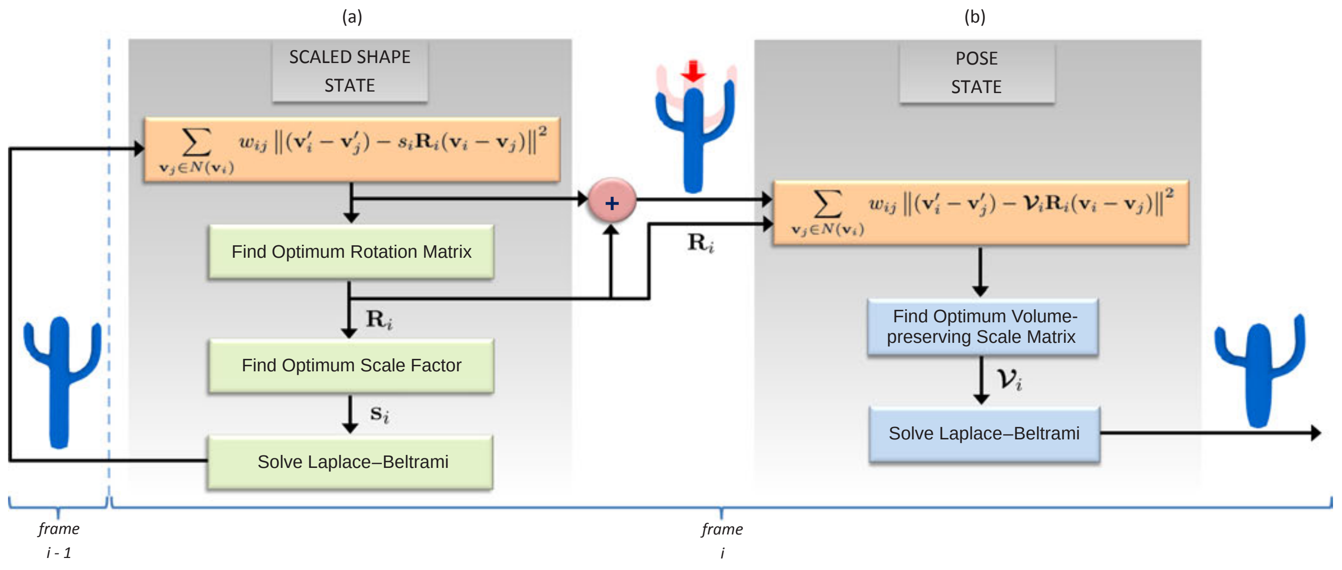

(a)
(b)
SCALED SHAPE STATE
POSE STATE
$\sum_{\mathbf{v}_j \in N(\mathbf{v}_i)} w_{ij} \left\|(\mathbf{v}'_i - \mathbf{v}'_j) - s_i \mathbf{R}_i(\mathbf{v}_i - \mathbf{v}_j)\right\|^2$
Find Optimum Rotation Matrix
$\mathbf{R}_i$
Find Optimum Scale Factor
$\mathbf{s}_i$
Solve Laplace–Beltrami
+
$\mathbf{R}_i$
$\sum_{\mathbf{v}_j \in N(\mathbf{v}_i)} w_{ij} \left\|(\mathbf{v}'_i - \mathbf{v}'_j) - \mathcal{V}_i \mathbf{R}_i(\mathbf{v}_i - \mathbf{v}_j)\right\|^2$
Find Optimum Volume-preserving Scale Matrix
$\mathcal{V}_i$
Solve Laplace–Beltrami
frame i - 1
frame i

Figure 4: *Our surface-deformation approach consists of two states. (a) In the preliminary scaled-shape state, overlapping cells covering the mesh surface are rotated and scaled uniformly to satisfy the given constraints. (b) The intermediary surface with its cells rotated but not scaled is then used in the pose state to achieve the volume-preserved end result. While the shape state can be run continuously for finer minimization of the deformation energy function (Equation 4), the pose state is visited only once per frame.*

preservation of edge lengths. Considering this goal, we developed a two-state deformation approach (Figure 4).

Initially, we use a *scaled-shape state*, where the surface is covered with overlapping cells such that uniform scale coefficients are used for cells as well as rotations. Therefore, cells can be rotated and scaled uniformly to satisfy given constraints. The main contribution of the scale coefficient is making squash-and-stretch effects possible with volume preservation by canceling rotations where only scale can satisfy the given constraints. Incorporating the scale component into ARAP deformation energy [SA07] yields:

$$\sum_{\mathbf{v}_j \in N(\mathbf{v}_i)} W_{ij} \left\|(\mathbf{v}'_i - \mathbf{v}'_j) - s_i \mathbf{R}_i(\mathbf{v}_i - \mathbf{v}_j)\right\|^2 . \tag{4}$$

In this energy function, there are three unknowns: the new positions of the vertices ($\mathbf{v}'$); the rotation matrix ($\mathbf{R}_i$) and the scale component (s_i). This minimization problem is solved by developing a three-step optimization approach. In the first step, the method finds the initial guess for the surface by solving Equation (3) or using the surface in the previous frame. The initial guess is then used to fill $(\mathbf{v}'_i - \mathbf{v}'_j)$, which leaves $\mathbf{R}_i$ as the only unknown while s_i is treated as constant. Given the covariance matrix M_i,

$$\mathbf{M}_i = \sum_{\mathbf{v}_j \in N(\mathbf{v}_i)} W_{ij}(\mathbf{v}_i - \mathbf{v}_j)(\mathbf{v}'_i - \mathbf{v}'_j)^T , \tag{5}$$

$\mathbf{R}_i$, which minimizes Equation (4), can be found using the singular value decomposition (SVD) of $\mathbf{M}_i$ [Sor09]:

$$\mathbf{R}_i = \mathbb{V}_i \mathbb{U}_i^T \ \textit{where}\ \mathbf{M}_i = \mathbb{U}_i \Sigma_i \mathbb{V}_i^T . \tag{6}$$

Note that $det(\mathbf{R}_i)$ should be guaranteed to be positive by changing the sign of the eigenvector in $\mathbf{U}_i$, which corresponds to the smallest eigenvalue in such a case.

In the second step, by concatenating edge vectors and rewriting the deformation energy of a vertex $\mathbf{v}_i$, where $\mathbf{e}'_{ij} = \sqrt{w_{ij}}(\mathbf{v}'_i - \mathbf{v}'_j)$, $\mathbf{e}_{ij} = \sqrt{w_{ij}}(\mathbf{v}_i - \mathbf{v}_j)$, results in:

$$\left\| \begin{bmatrix} \mathbf{e}'_{ij} \\ \mathbf{e}'_{ik} \\ \vdots \end{bmatrix} - s_i \begin{bmatrix} \mathbf{R}_i \mathbf{e}_{ij} \\ \mathbf{R}_i \mathbf{e}_{ik} \\ \vdots \end{bmatrix} \right\|^2 \rightarrow \left\| \mathbf{e}'_i - s_i {\mathbf{e}^r}_i \right\|^2 . \tag{7}$$

Then, the scale factor for cell i in the scaled-shape state can be found by setting the derivative of Equation (7) to zero:

$$s_i = \frac{{\mathbf{e}'_i}^T {\mathbf{e}^r}_i}{{{\mathbf{e}^r}_i}^T {\mathbf{e}^r}_i} . \tag{8}$$

In the third step, the computed rotation matrices and scale coefficients are placed into the partial derivative of Equation (4) with respect to $\mathbf{v}'$ by treating the rotation matrices and scale coefficients as constants. To find $\mathbf{v}'$, which minimizes the deformation energy, setting the derivative to zero results in:

$$\sum_{\mathbf{v}_j \in N(\mathbf{v}_i)} W_{ij}(\mathbf{v}'_i - \mathbf{v}'_j) = \sum_{\mathbf{v}_j \in N(\mathbf{v}_i)} w_{ij} \frac{(s_i \mathbf{R}_i + s_j \mathbf{R}_j)}{2} (\mathbf{v}_i - \mathbf{v}_j). \quad (9)$$

One can see that the left-hand side of this equation corresponds to Equation (1), so we can fill δ using the right-hand side of the equation, which is all known in our system. After that, Equation (3) can be solved to find $\mathbf{V}'$. This optimization can be pursued for further refinement by reiterations of the three-step cycle. The intermediary surface with its cells rotated but not scaled is called the *pose state*, and it simulates volume preservation as described below.

Simulating Volume Preservation. Several approaches for preserving volume while deforming the mesh, several approaches have been proposed in the literature. In Zhou *et al.*'s work [ZHS*05], a volumetric graph of the mesh is constructed and the deformation energy function is augmented to preserve the approximate volume and surface details. In Huang *et al.*'s approach [HSL*06], the volume of the mesh is integrated as a non-linear hard constraint into the energy function so that volume can be preserved exactly. We only try to simulate volume preservation without guaranteeing any exactness. Inspired by [CPIS02], we follow an approach where the volume of the mesh is altered by applying a scale matrix

$$\mathbf{S}_s = \begin{bmatrix} s & 0 & 0 \\ 0 & \sqrt{1/s} & 0 \\ 0 & 0 & \sqrt{1/s} \end{bmatrix} \quad (10)$$

in the deformation coordinate system, where the x-axis is the primary axis on which squash-stretch effects take place. We extend this concept into the vertex level, where each vertex has its own scale matrix. The deformation coordinate system's orientation of each vertex is dynamically adjusted by considering the deformation of the vertex's one-ring neighbours. For this purpose, we find the principal components by applying an eigenvalue decomposition on the covariance matrix of the edge differences:

$$\sum_{j \in N(i)} (\mathbf{e}'_{ij} - \mathbf{R}_i \mathbf{e}_{ij})(\mathbf{e}'_{ij} - \mathbf{R}_i \mathbf{e}_{ij})^T = \mathbf{P}_i \Sigma_i \mathbf{P}_i{}^T, \quad (11)$$

where $\mathbf{P}_i$ contains eigenvectors sorted in decreasing order of their eigenvalues. The largest eigenvector is used as the primary axis for scale because it represents the most significant deformation axis for the one-ring neighbours. Having $\mathbf{P}_i$, which represents the deformation coordinate system for vertex $\mathbf{v}_i$, we can obtain the volume-preserving scale matrix $\mathbf{V}_i$ by first projecting on $\mathbf{P}_i$, applying scale matrix $\mathbf{S}_{s_i}$ and then reprojecting to the original coordinate system:

$$\mathcal{V}_i = \mathbf{P}_i \mathbf{S}_{s_i} \mathbf{P}_i^T, \quad (12)$$

where s_i is the scale coefficient calculated using Equation (8), with the projected edges on the primary axis. Our method is also applicable when it is desired to have volumetric changes exaggerated. This volumetric effect is easily achieved by increasing or decreasing s_i with a factor λ_v, using a linear equation: $s'_i = (s_i - 1)\lambda_v + 1$.

The volume-preserving scale matrices found in this step are used in the second state of the system (Figure 4), which is periodically executed to obtain the pose. In this state, while calculating the right-hand side for Equation (9), rotations are pre-multiplied by the corresponding volume matrices instead of by scale coefficients:

$$\sum_{\mathbf{v}_j \in N(\mathbf{v}_i)} W_{ij} \left\| (\mathbf{v}'_i - \mathbf{v}'_j) - \mathbf{V}_i \mathbf{R}_i (\mathbf{v}_i - \mathbf{v}_j) \right\|^2. \quad (13)$$

There are several numerical techniques for solving the sparse linear system of Equation (3) [BS08]. We follow an approach where Equation (3) is considered a least-squares problem, $min \; \|\mathbf{Ax} - \mathbf{b}\|^2$, with Cholesky factorization applied on the normal matrix $\mathbf{A}^T\mathbf{A}$, in the normal equation, $\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}$ [LSCo*04]. This factorization can then be used multiple times to solve this equation for different right-hand sides, with only the cost of back substitution. Note that adding (or removing) soft constraints into the system only affects the diagonal of the normal matrix $\mathbf{A}^T\mathbf{A}$, which is always non-zero. Therefore, with refactorization, we can skip computing the permutation matrix, which is used for reducing fill-ins, because the non-zero pattern of the matrix is not changed. This improvement becomes particularly important in interactive editing systems like ours, where the user frequently adds or removes control points and expects an immediate response.

5. Trajectory Retargeting

With the pose-to-pose examples of the input skeleton and the mesh model, our solution tries to learn a mapping between a joint and corresponding control point for trajectory retargeting. We consider the following to be desirable characteristics of our mapping function:

1. The mapping function should be translation and rotation invariant (more precisely, applying translation and rotation to example poses and the test case should have a minimal effect on the final retargeting result).
2. Retargeted trajectories should be smooth (i.e. no jitter or jerkiness).
3. Given a joint position example as an input, the mapping result should be the corresponding control point example.
4. The mapping function should be suitable for integrating user- or system-specified constraints.

Affine Combination with Bounded Weights. To meet the above requirements, we first consider an alternative *affine combination* approach for the mapping function and then discuss our method. Assume that for a joint $\mathbf{j}$, we have n joint-control point examples $\mathbf{j}^{(1)} \leftrightarrow \mathbf{c}^{(1)}, \ldots, \mathbf{j}^{(n)} \leftrightarrow \mathbf{c}^{(n)}$. Given an animation frame k, with the position of joint $\mathbf{j}$ as $\mathbf{j}_k$, the mapping function is responsible for finding the corresponding control point location $\mathbf{c}_k$. In this approach, $\mathbf{j}_k$ is represented in terms of $\mathbf{j}^{(1)}, \ldots, \mathbf{j}^{(n)}$, using bounded weights with an affine combination, by solving a constrained linear system of

$$\begin{bmatrix}\begin{bmatrix}\mathbf{j}^{(1)}\\ 1\end{bmatrix} \cdots \begin{bmatrix}\mathbf{j}^{(n)}\\ 1\end{bmatrix}\end{bmatrix}\begin{bmatrix}w_1\\ \vdots\\ w_n\end{bmatrix} = \begin{bmatrix}\mathbf{j}_k\\ 1\end{bmatrix}, \tag{14}$$

so that the computed weights are applied to the corresponding control points $\mathbf{c}^{(1)}$, ..., $\mathbf{c}^{(n)}$ in order to find $\mathbf{c}_k$:

$$\sum_{i=1}^{n} w_i \begin{bmatrix}\mathbf{c}^{(i)}\end{bmatrix} = \begin{bmatrix}\mathbf{c}_k\end{bmatrix}. \tag{15}$$

Using an affine combination of poses provides translation invariance, satisfying the first characteristic of our desired mapping function. This technique is employed by Baran *et al.* [BVGP09], who use a weighting schema for example poses that are represented by a vector in shape space. Although translation invariance is enforced as a soft constraint in our system, slight violations do not affect the retargeting results significantly. Note that because weights are not bounded in a range in these approaches, there is the advantage of having no limit for extrapolation. On the other hand, too-large weights are not meaningful and may easily lead to unnatural results. Bregler *et al.* [BLCD02] also put a margin on weights to limit extrapolation while finding affine combinations of example shapes in their work.

In addition to the fact that this approach can only handle the first and second characteristics (i.e. translation invariance and smoothness), there is a further shortcoming. If more than four independent examples are provided for a control point, the system becomes underdetermined and there is no exact way to select the proper solution from among many solutions.

Spacetime Solution. Our spacetime solution addresses the shortcomings of the affine combination approach and allows direct mapping and addition of constraints. We also handle possible jitter and jerkiness by introducing an acceleration constraint as a soft constraint. Considering all F frames together, the function to be minimized is

$$\min_{\mathbf{w}} \sum_{i\in F} \|\mathbf{J}_i\mathbf{w}_i - \mathbf{j}_i\|^2, \tag{16}$$

where $\mathbf{J}_i$ represents a 4×4 matrix filled with selected example joint positions for frame number i with soft affine constraints (Equation 14). The corresponding linear system for this equation can be constructed as

$$\min_{\mathbf{w}} \left\|\mathcal{J}\mathcal{W} - j\right\|^2, \tag{17}$$

with

$$\mathcal{J} = \begin{bmatrix}\mathbf{J}_1 & \cdots & \mathbf{0}\\ \vdots & \ddots & \\ \mathbf{0} & & \mathbf{J}_F\end{bmatrix}, \mathcal{W} = \begin{bmatrix}\mathbf{w}_1\\ \vdots\\ \mathbf{w}_F\end{bmatrix}, \mathcal{J} = \begin{bmatrix}\mathbf{j}_1\\ \vdots\\ \mathbf{j}_F\end{bmatrix}. \tag{18}$$

In the same manner, we use the example control point positions for finding the mapped trajectory:

$$\mathcal{C}w = c,\ \mathcal{C} = \begin{bmatrix}\mathbf{C}_1 & \cdots & \mathbf{0}\\ \vdots & \ddots & \\ \mathbf{0} & & \mathbf{C}_F\end{bmatrix}, \boldsymbol{c} = \begin{bmatrix}\mathbf{c}_1\\ \vdots\\ \mathbf{c}_F\end{bmatrix}, \tag{19}$$

where $\mathbf{C}_i$ represents a matrix filled with example control point positions corresponding to selected example joint positions for frame number i, and $\mathbf{c}$ is the result of the trajectory mapping.

Constraints. Our approach also differs from the previous approaches in its solution to integrating constraints into the spacetime system. Because our weights might belong to different examples for adjacent frames, we cannot directly apply temporal or equality constraints on the weights. To overcome this problem, we first obtain the mapped trajectory from the calculated weights, and then minimize the acceleration along the trajectory:

$$\min_{\mathbf{w}} \sum_{i=2}^{F-1} \|\mathbf{C}_{i-1}\mathbf{w}_{i-1} - 2\mathbf{C}_i\mathbf{w}_i + \mathbf{C}_{i+1}\mathbf{w}_{i+1}\|^2. \tag{20}$$

The corresponding linear system and the acceleration matrix are thus:

$$\min_{w} \|\mathcal{A}\mathcal{C}w\|^2,\ \mathcal{A} = \lambda_A \begin{bmatrix}0 & & & & \\ \mathbf{I} & -2\mathbf{I} & \mathbf{I} & & \\ & & \ddots & & \\ & & \mathbf{I} & -2\mathbf{I} & \mathbf{I}\\ & & & & 0\end{bmatrix}, \tag{21}$$

where λ_A is the weight of the acceleration constraint and the only parameter for our trajectory retargeting system.

Regarding linear constraints, we provide support for position and velocity constraints as hard constraints:

$$\mathbf{C}_i\mathbf{w}_i = \mathbf{p}_i \tag{22}$$

$$\text{and}\quad \mathbf{C}_i\mathbf{w}_i - \mathbf{C}_{i-1}\mathbf{w}_{i-1} = \mathbf{v}_i, \tag{23}$$

where the position and velocity of the control point at frame i are constrained to $\mathbf{p}_i$ and $\mathbf{v}_i$, respectively.

Spacetime System. Integrating position and velocity constraints as hard constraints and the acceleration constraint as a soft

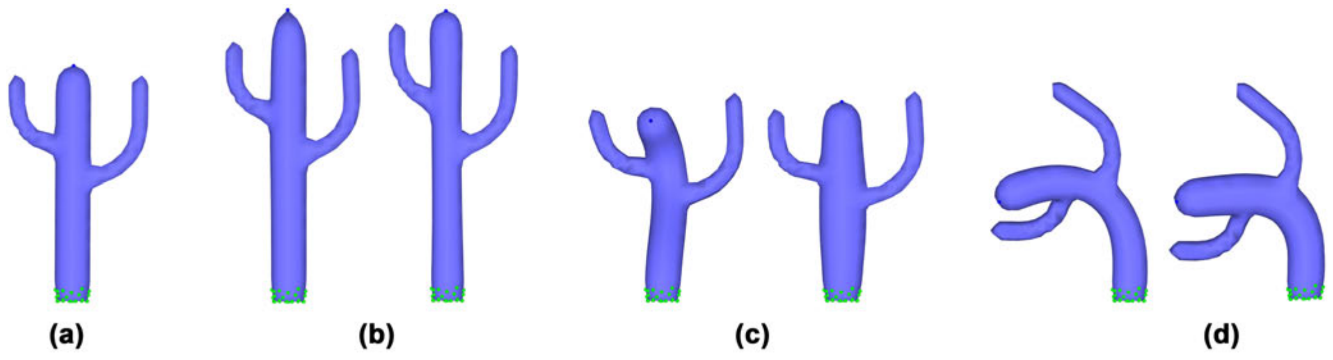


Figure 5: *Comparison of rigid and squash-and-stretch deformations [SA07]. Image (a) is the original model; Images (b–d) show ARAP deformation results on the left and our deformation results on the right. Because the nature of ARAP deformation relies on edge preservation, constraints that implicitly require non-rigid deformations (e.g. b–c) lead to more rigid results.*

constraint, we can construct a linear system with equality constraints:

$$\min_{w} \left\| \begin{bmatrix} \mathcal{J} \\ \mathcal{A}C \end{bmatrix} w - \begin{bmatrix} j \\ \mathbf{0} \end{bmatrix} \right\|^2, \ \mathcal{H}Cw = \hbar, \tag{24}$$

where each position and velocity constraint represents a row in matrix $\boldsymbol{H}$, and each associated constraint value is placed under $\mathbf{h}$. As our formulation involves with only equality constraints, we can use Lagrange multipliers to obtain a single linear system of

$$\begin{bmatrix} \begin{bmatrix} \mathcal{J} \\ \mathcal{A}C \end{bmatrix}^T \begin{bmatrix} \mathcal{J} \\ \mathcal{A}C \end{bmatrix} & C^T\mathcal{H}^T \\ \mathcal{H}C & 0 \end{bmatrix} \begin{bmatrix} w \\ \lambda \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \mathcal{J} \\ \mathcal{A}C \end{bmatrix}^T \begin{bmatrix} j \\ \mathbf{0} \end{bmatrix} \\ \hbar \end{bmatrix}. \tag{25}$$

Although we do not bound weights in a range in our current implementation, that can easily be achieved by solving Equation (24) with extra inequality constraints, along with the existing equality constraints, using quadratic programming for the solution.

We can also combine systems generated for different control points into one system, and introduce constraints between control points in a very similar manner to what we did for previous constraints. A useful application for this feature is generating an equality constraint on the y-axis for two control points at a switch frame, when the contact anchor is transferred from one to another.

Adding Global Position. Since the result of trajectory retargeting only includes motion in the local coordinate frame of the mesh, we use a simple technique to create an appropriate global position, particularly for motions that include locomotion or that accommodate flight (such as jumping, hopping, leaping). We define *switch frames* as frames at which the source motion switches from the contact to the flight phase. At each switch frame, the angular and linear velocity of the mesh is calculated. Then, a physical simulation of the mesh is activated until the mesh reaches the ground. Next, the contact phase is activated, where the closest control point to the ground is used as an anchor, and local movement of this point is reflected to the global position of the mesh. We manually extract switch frames from the source motion, but automatic techniques [LCR*02] can also be used.

6. Results

We implement our solution using C++ on an Intel 2.6 GHz Quad-Core laptop with 8 GB RAM. For linear algebra computations, we use the Eigen library, and for linear systems with boundary constraints, we use the quadratic programming implementation provided by Matlab's optimization toolbox.

Mesh Deformation. We first compare our deformation results with the ARAP deformation [SA07] and the volume-preserving deformation [HSL*06] methods. Our Supporting Information Video S1 shows a comparison of the deformation results, and Figure 5 illustrates the comparison of our method and the ARAP method with two example cases. Rigid deformations cannot properly handle cases where control point constraints impose changes in edge lengths (e.g. squash and stretch). We also compare our results with Huang *et al.*'s. [HSL*06], in which volume constraint has a global effect and is satisfied regardless of the deformed parts. In other words, squashing the mesh can create a balloon-like effect, where any part of the mesh can swell. Our algorithm's volume preservation results are local to deformed parts, thus the remaining parts are not affected by volume-related changes. Figure 6 presents a comparison of the two methods.

Table 1 shows a comparison of an unoptimized implementation of our method with ARAP deformation in terms of performance. Our measure of performance consists of two main components: performance of a single iteration Δ_i and the required number of iterations N_i for an acceptable convergence. For each frame, the ARAP method contains three iterations and our method contains two iterations from Figure 4(a) and one iteration from (b). As expected, performance results are comparable in terms of Δ_i because both methods employ SVD computation for rotations and sparse linear system solution for the Laplace–Beltrami operator. For a coarse

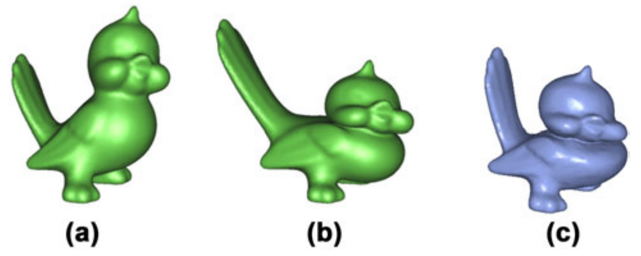


Figure 6: *Comparison of our model with Huang et al.'s. [HSL*06]. (a) The original model. (b) Huang et al.'s deformation result with volume preservation. (c) Our deformation result. Notice that, because volume preservation is a global effect in [HSL*06], squashing the mesh creates a balloon-like effect, where the tail of the bird swells. Our volume-preservation results are local to deformed parts, and hence the tail remains unchanged.*

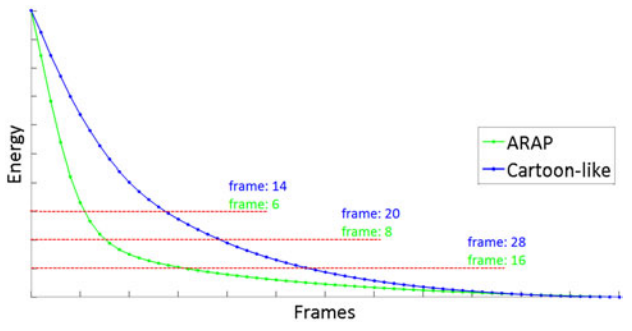


Figure 7: *N_i—Change rates of energy functions in the first 60 frames for given constraints (repeated for several different constraints). The red lines show frame numbers when energy functions drop to 30%, 20% and 10%. Similarly, we calculate average frame numbers between 30% and 5%, by sampling each 1% drop, as a representative of N_i.*

comparison of N_i, we provide the change rates of energy functions in ARAP deformation [SA07] and in Equation (4) (Figure 7). As we introduce a new variable for the scale coefficient in our equation, the change rates of our function are slower than the ARAP method's. As a result, if we assume that a decrease in energy function between 30% and 5% is an acceptable convergence range, the performance ratio of our and the ARAP method is 2.4 in terms of $\Delta_i N_i$, on average.

Motion Retargeting. To demonstrate the abilities of our method, we use three well-known mesh models (Figure 1) with several motion-capture sequences obtained from CMU Graphics Lab Motion Capture Database [CMU14]. Video S1 demonstrates the retargeting results for the cactus, T-Rex and paper models.

On our hardware, retargeting a motion-capture sequence of 3700 frames takes on average 73 s when eight joint-control point examples are chosen. For 12 examples, the runtime for the same sequence extends to 161 s on average.

The only possible parameter for our retargeting system is λ_A. Larger values may result in losing high-frequency details in retargeted motion, making the models look lifeless; smaller values might fail to reduce jitter and thus might create unrealistic results. We set λ_A to 0.3 for all our examples in this work, and demonstrate its effect in the video material.

Table 1: *Δ_i–Deformation performance statistics, demonstrating the performance comparison between the deformations produced by the ARAP method and our method in frames per second (fps).*

Model	No. vertices	ARAP	Our method
Cactus	620	86.1	69.7
T-Rex	5621	9.0	7.5
Armadillo	10 488	4.1	3.5
Bird	5302	8.9	7.6

We also use our models as inputs for the work of Baran and Popović [BP07] and the Mixamo rigging and animation tool. Note that it is not exactly fair to compare our results with these methods' results because both methods require no training, and because the target mesh models are considered to be humanoid models. However, our aim is to investigate how automatic approaches developed for humanoid structures work on such models rather than to compare these methods directly. Figure 8 illustrates the results of Baran and Popović method [BP07]. The most satisfactory results are achieved for the legs of the T-Rex model, owing to its similarity to humanoid legs; however, the Mixamo tool failed to rig the cactus model because of the absence of legs. Although rigging is achieved for the T-Rex model, the produced animation is not satisfactory in terms of quality and end effector placements.

6.1. User study

To evaluate the authoring capability of our retargeting system, we performed a qualitative usability test. We adopted the *case-study–based usability evaluation* technique from among alternative usability evaluation techniques [LFH10]. Because the authoring task is a difficult problem to evaluate numerically due to the absence of quantitative metrics such as task performance or efficiency, a qualitative data analysis method was most suitable for our purpose.

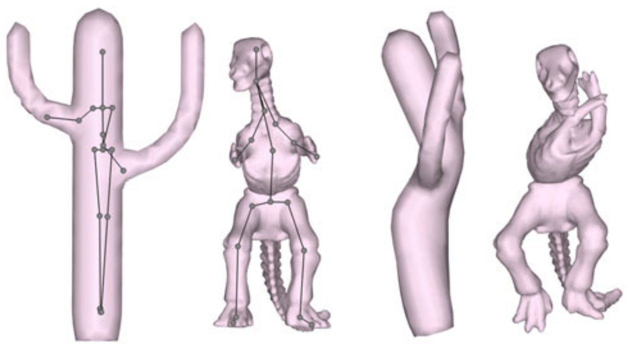


Figure 8: *Rigging and animation results using [BP07] for the jumping motion.*

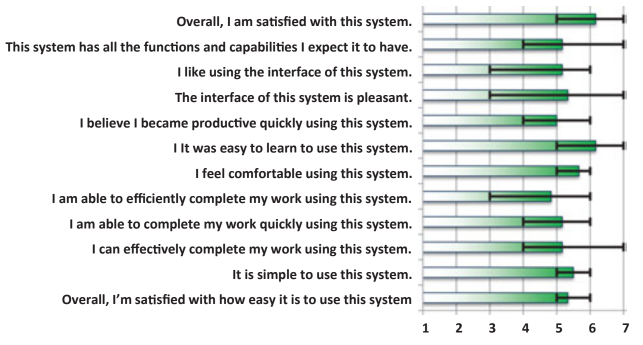
Method. Eight users participated in our tests, ranging in age from 22 to 47 years, and all had intermediate- to advanced-level knowledge of popular 3D mesh-deformation/modelling tools. Before the test, the participants first had a training phase in which they practiced the steps of our method and created simple animations. After they were comfortable with using the system, the task for the main test was given. The tests took place in sessions of approximately 1 h for each participant, including the training phase. Total interaction time reduces considerably as a user becomes more experienced with the system, e.g. authoring for a 2-min long sequence usually takes 5–10 min.

To collect data about the participants' interactions with our system, we adopted the direct observation [LFH10], video protocol [WM91] and thinking-aloud [Lew82] techniques. During the sessions, participants were encouraged to elaborate on problems they encountered and techniques they used to recover from mistakes, and to provide comments and/or suggestions for improvement.

Observations. Our user study video summarizes the critical observations captured during participants' test sessions (Video S2). Upon reviewing the recordings of the test sessions, we observed several patterns in user behaviour:

- *Mesh Control Point Selection.* During the selection of joint-control point examples, all participants began by choosing control points around extreme locations of the mesh models (e.g. corners in the paper model). It is possible to generalize that users conceptually give higher importance to the extremities of the mesh models.
- *Symmetrical Control Points.* One of the most commonly desired features is the symmetrical control point marking aspect, which allows the system to automatically double the training data by simply using the symmetry plane. Another benefit of symmetrical marking is that generating symmetrical training data leads to symmetrical results in retargeted motion, which is a generally desired property.
- *Skeletal Joint Selection.* It is observed that all users began joint selection by matching the end effectors, i.e. both hands and both feet, to their desired control point locations on the mesh as joint-control point examples. Following this matching, most users finished the process after selecting only a few more joints, whereas some users extended the selection to nearly all joints. The selected number of joints ranged from 7 to 9 for most users, to the extreme values of 13 and 15 for two users. We believe that particularly the expert users tend to select a large number of joints, even though they were instructed that it is not necessary to select that many.
- *Key Pose Selection.* Users consistently made sensible choices regarding pose-to-pose examples, to the extent that major extreme poses of the input motion were usually covered. For example, in the walking motion, the frame where the character's swing foot hits the ground was selected as the key pose by most users.
- *Constraints.* Another useful feature of our system is its constraint generation for the retargeting process, which can be done by the user for a case where the constraint position does not represent a generic correspondence (hence is not suitable for integrating in training data) but is specific to a single frame. Alternatively, the system can automatically introduce constraints, for example, to force control points to have the same y-axis value when the contact anchor is transferred from one to another.

Table 2: *The CSUQ Questionnaire Results. The participants were asked to evaluate each statement by a score between 1, 'strongly disagree', and 7, 'strongly agree'. The bar indicates the mean score for the statement and the error bar indicates the range of scores given.*



Test and Usability Questionnaire Feedback. Following the test, we asked the participants to evaluate the usability of our system, using the CSUQ method [Lew95].

The questionnaire feedback (Table 2) shows that, after a training session, participants were comfortable and capable of performing the authoring task (5.67/7). The responses confirm that our authoring system is simple to use (5.50/7) and easy to learn (6.17/7). Participants also generally agreed that they felt productive while using the system (5.00/7), and that the system provided all the functionalities they expected it to have (5.17/7). The results also show that the participants felt marginally that the system is efficient (4.83/7). In general, the responses show that the participants were satisfied with the system (6.17/7).

Also, it was a repeated opinion during the tests that the deformations not only in the produced animations but also in the immediate results during mesh-posing were appealing and facilitated easy authoring. Overall, participants shared the opinion that the produced animations were of sufficient quality and could be further improved with ease by post-processing using professional authoring tools.

Problems and Suggestions. With the help of the user study, we were able to identify problems and issues with our tool that need to be addressed in future work:

- No matter how non-humanoid the mesh was, many users tended to attribute a significant degree of false anthropomorphism to the target mesh models during posing. For example, dragging the front legs of the T-Rex model down, to match its pose to an input human key pose with resting arms was a common mistake.
- The lack of colour-coded transformation manipulators, which are usually available with professional authoring tools, was an issue raised by three users. In our system, the skeleton and the mesh are controlled independently in different viewports. These users

Table 3: *The viewer study results. The preference score, defined as the mean of viewer ratings, for each animation and the average of preference scores with respect to the number of joint-control point examples used are shown in both evaluation criteria. Each animation was retargeted with selecting only four pose-to-pose examples.*

Input motion / Target mesh	Mickey-Walk / Armadillo	Directing traffic / Cactus	Boxing / Paper	Average
	Similarity			
8 examples	3.1	3.4	3.4	3.30
10 examples	3.8	3.9	2.9	3.53
12 examples	3.3	3.2	—	3.25
	Overall Quality			
8 examples	3.0	3.3	3.3	3.20
10 examples	3.8	3.6	3.2	3.53
12 examples	3.1	3.2	—	3.15

reported that the absence of such manipulators creates confusion, particularly in the paper and the cactus models, to differentiate the front and back regions of the mesh. A suggestion was to autorotate one viewport when the other is rotated, to have a better mapping between the two.

- It was stated by the users that while they were editing a mesh pose, the previous poses were invisible, making it difficult to establish correspondence between the previous and the current key pose. To resolve this issue, we added the capability to display the silhouette of the previous key pose of the mesh in our system's interface.
- Users reported that they would benefit from visual feedback to prevent self-collisions of the mesh while posing.

6.2. Viewer study

We asked 10 viewers (different from the user study group) to evaluate a total of eight mesh animations that were realized by our system. The viewers were asked to rate each animation between 1 and 5 in terms of similarity to the input motion-capture sequence, where 1 and 5 represent 'very dissimilar' and 'very similar', respectively, and overall quality, where 1 and 5 represent 'very bad' and 'very good', respectively. The results are summarized in Table 3.

It is seen that the number of joint-control point examples can significantly affect the system outcome. However, there is no direct correlation between this number and either of the two evaluation criteria. While the results of the armadillo and the cactus models scored the highest with 10 examples and the lowest with 12 examples, the result of the paper model with 8 examples were in general preferred over the one with 10 examples. Thus, we can conclude that the morphology of the mesh model dictates the ideal set of joint-control point examples.

7. Discussion and Conclusion

We present a technique for retargeting motions from humanoid skeletons to arbitrary meshes. Our method is suitable for generating squash-and-stretch effects by providing volume-preserving deformations in the range between rigid and cartoon-like, and for retargeting human motions to arbitrary mesh models using example poses provided by the user.

One limitation of our solution is its restricted ability to generate global position for the target mesh after trajectory retargeting. While our system is capable of generating plausible global trajectories for simple motions, more advanced post-processing steps can be used, as in [YAH10]. Similarly, our results could be improved with the help of refinements towards physical realism and by a more advanced global transformation generation system. Another limitation of our method is that a fully articulated deformation of the mesh model is not possible. This problem is inherited from surface-based deformation approaches, where no extra information about articulation of the mesh is considered. Automatic mesh segmentation algorithms could be used for extracting articulation information by partitioning the surface into segments. Then, an augmented deformation system, where the rigidity of the segments is higher than the rigidity of the segment boundaries, could provide articulated deformation. Future work could also address the support of multi-component models.

More improvements can be made to lessen or simplify required user interactions. As our method allows the user to determine example poses, our assumption is that the user provides a sufficient set of extreme poses to represent the source motion. It might be possible to suggest a set of key skeleton poses by pre-processing the source motion and extracting a few poses that are as far as possible from each other, as in [BVGP09]. Another improvement could be to estimate the number of example poses from a given source motion, which would balance the trade-off between the retargeting quality and the amount of user interaction.

Acknowledgement

This work is supported by the Scientific and Technical Research Council of Turkey (TUBITAK, project number 112E105).

References

[BLCD02] Bregler C., Loeb L., Chuang E., Deshpande H.: Turning to the masters: Motion capturing cartoons. In *Proceedings of SIGGRAPH '02* (New York, NY, USA, July 2002), vol. 21, ACM, pp. 399–407.

[BP07] Baran I., Popović J.: Automatic rigging and animation of 3D characters. In *Proceedings of SIGGRAPH '07* (New York, NY, USA, July 2007), vol. 26, ACM.

[BS08] Botsch M., Sorkine O.: On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (January 2008), 213–230.

[BTST12] Bharaj G., Thormahlen T., Seidel H.-P., Theobalt C.: Automatically rigging multi-component characters. *Computer Graphics Forum 31*, 2pt3 (May 2012), 755–764.

[BVGP09] BARAN I., VLASIC D., GRINSPUN E., POPOVIĆ J.: Semantic deformation transfer. In *Proceedings of SIGGRAPH '09* (New York, NY, USA, July 2009), vol. 28, ACM, pp. 36:1–36:6.

[CMU14] Carnegie-Mellon University Graphics Lab Motion Capture Database. http://mocap.cs.cmu.edu. Accessed January 2014.

[CPIS02] CHENNEY S., PINGEL M., IVERSON R., SZYMANSKI M.: Simulating cartoon style animation. In *NPAR '02: Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2002), ACM, pp. 133–138.

[Gle98] GLEICHER M.: Retargetting motion to new characters. In *Proceedings of SIGGRAPH '98* (New York, NY, USA, 1998), ACM, pp. 33–42.

[HRE*08] HECKER C., RAABE B., ENSLOW R. W., DEWEESE J., MAYNARD J., PROOIJEN K. V.: Real-time motion retargeting to highly varied user-created morphologies. In *Proceedings of SIGGRAPH '08* (2008), vol. 27.

[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. In *Proceedings of SIGGRAPH '06* (New York, NY, USA, 2006), ACM, pp. 1126–1134.

[IAF09] IKEMOTO L., ARIKAN O., FORSYTH D.: Generalizing motion edits with Gaussian processes. In *Proceedings of SIGGRAPH '09* (New York, NY, USA, 2009), vol. 28, ACM, pp. 1:1–1:12.

[JMD*07] JOSHI P., MEYER M., DEROSE T., GREEN B., SANOCKI T.: Harmonic coordinates for character articulation. In *Proceedings of SIGGRAPH '07* (New York, NY, USA, July 2007), vol. 26, ACM.

[LCR*02] LEE J., CHAI J., REITSMA P. S. A., HODGINS J. K., POLLARD N. S.: Interactive control of avatars animated with human motion data. In *Proceedings of SIGGRAPH '02* (New York, NY, USA, July 2002), vol. 21, ACM, pp. 491–500.

[Lew82] LEWIS C., : *Using the'thinking-aloud' method in cognitive interface design*. IBM TJ Watson Research Center, 1982.

[Lew95] LEWIS J. R.: IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *International Journal Human-Computer Interaction 7*, 1 (January 1995), 57–78.

[LFH10] LAZAR J., FENG J. H., HOCHHEISER H.: *Research Methods in Human-Computer Interaction*. Wiley Publishing, Hoboken, NJ, USA 2010.

[LLCO08] LIPMAN Y., LEVIN D., COHEN-OR D.: Green coordinates. In *Proceedings of SIGGRAPH '08* (New York, NY, USA, August 2008), vol. *27*, ACM, pp. 78:1–78:10.

[LSCo*04] LIPMAN Y., SORKINE O., COHEN-OR D., LEVIN D., RÖSSL C., PETER SEIDEL H.: Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 181–190.

[LSLCO05] LIPMAN Y., SORKINE O., LEVIN D., COHEN-OR D.: Linear rotation-invariant coordinates for meshes. In *Proceedings of SIGGRAPH '05* (New York, NY, USA, 2005), ACM, pp. 479–487.

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *SGP '07: Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 109–116.

[SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (New York, NY, USA, 2004), ACM, pp. 175–184.

[SK04] SHEFFER A., KRAEVOY V.: Pyramid coordinates for morphing and deformation. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 68–75.

[sMBBT00] SÉBASTIEN MONZANI J., BAERLOCHER P., BOULIC R., THALMANN D.: Using an intermediate skeleton and inverse kinematics for motion retargeting. *Computer Graphics Forum 19* (2000), 11–19.

[Sor09] SORKINE O.: Least-squares rigid motion using SVD. *Technical notes*, February (2009), 1–6.

[SZGP05] SUMNER R. W., ZWICKER M., GOTSMAN C., POPOVIĆ J.: Mesh-based inverse kinematics. In *Proceedings of SIGGRAPH '05* (New York, NY, USA, 2005), ACM, pp. 488–495.

[WM91] WRIGHT P. C., MONK A. F.: The use of think-aloud evaluation methods in design. *SIGCHI Bulletin 23*, 1 (January 1991), 55–57.

[XZY*07] XU W., ZHOU K., YU Y., TAN Q., PENG Q., GUO B.: Gradient domain editing of deforming mesh sequences. *ACM Transactions on Graphics 26*, 3 (July 2007), article no. 84.

[YAH10] YAMANE K., ARIKI Y., HODGINS J.: Animating non-humanoid characters with human motion data. In *SCA '10: Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2010), Eurographics Association, pp. 169–178.

[ZHS*05] ZHOU K., HUANG J., SNYDER J., LIU X., BAO H., GUO B., SHUM H.-Y.: Large mesh deformation using the volumetric graph Laplacian. In *Proceedings of SIGGRAPH '05* (New York, NY, USA, July 2005), vol. 24, ACM, pp. 496–503.

Supporting Information

Additional Supporting Information may be found in the online version of this article at the publisher's web site:

Video S1
Video S2