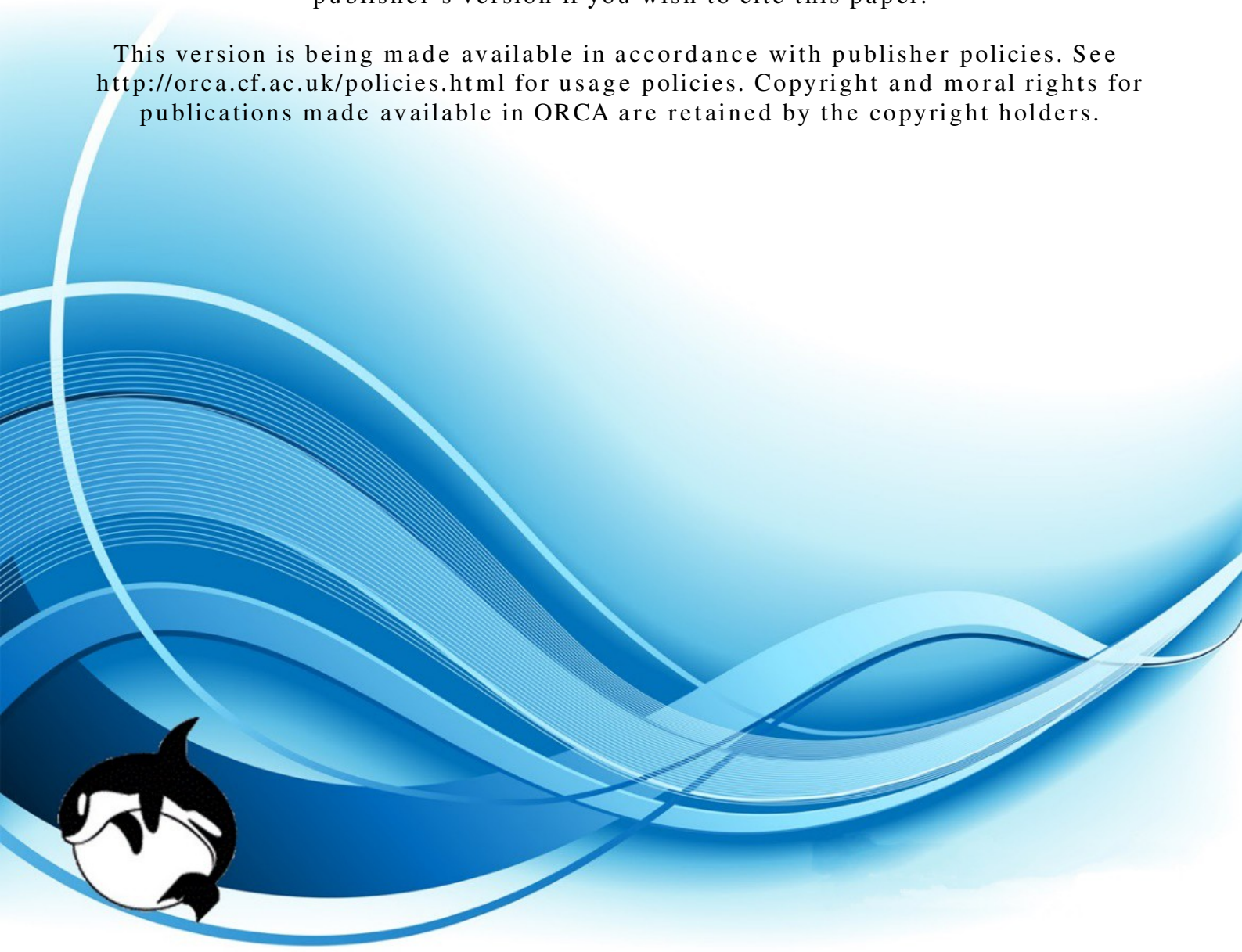


Biharmonic Deformation Transfer with Automatic Key Point Selection

Jie Yang[a,b], Lin Gao[a,*], Yu-Kun Lai[c], Paul L. Rosin[c], Shihong Xia[a,*]

[a]Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences
[b]University of Chinese Academy of Sciences
[c]School of Computer Science and Informatics, Cardiff University, UK

Abstract

Deformation transfer is an important research problem in geometry processing and computer animation.A fundamental problem for existing deformation transfer methods is to build reliable correspondences. This is challenging, especially when the source and target shapes differ significantly and manual labeling is typically used. We propose a novel deformation transfer method that aims at minimizing user effort. We adapt a biharmonic weight deformation framework which is able to produce plausible deformation even with only a few key points. We then develop an automatic algorithm to identify a minimum set of key points on the source model that characterizes the deformation well. While minimal user effort is still needed to specify corresponding points on the target model for the selected key points, our approach avoids the difficult problem of choosing key points. Experimental results demonstrate that our method, despite requiring little user effort, produces better deformation results than alternative solutions.

Keywords: shape deformation; biharmonic weights; key point selection; deformation transfer
2010 MSC: 00-01, 99-00

*Corresponding Author: gaolin@ict.ac.cn (Lin Gao) and xsh@ict.ac.cn (Shihong Xia)

1. Introduction

Shape deformation is a fundamental problem in computer animation and shape modeling. With the aim of generating realistic shapes, various approaches have been proposed, including skeleton rigging, shape deformation [1, 2] and deformation transfer [3, 4]. Skeleton rigging is suitable for shapes such as human bodies with a well-defined skeletal structure. Shape deformation is more flexible, but often requires specifying and moving a group of handles to produce a deformed shape. To produce a deformation sequence, it not only requires knowledge and expertise, but it is also tedious to produce each deformed shape.

When some deformed shapes are available, deformation transfer makes it possible to transfer the deformation of source shapes to target shapes, effectively reusing existing deformations. This makes it much more efficient to produce new deformed shapes, while avoiding the requirement of having shape deformation expertise. Previous work for deformation transfer mainly focuses on improving deformation transfer quality and extending it to handle general shapes and large deformation. Another key step for deformation transfer is finding reliable correspondences. However, this step is challenging, especially when the source and target shapes differ significantly (e.g. transferring the deformation of a human to an armadillo). In such cases, correspondences are either manually specified, or even if some semi-automatic algorithms are used, constraints of key correspondences are still required to be specified by the user. However, specifying a set of sufficient and effective correspondences requires expertise, including understanding of the underlying deformation transfer technique. In practice, this is often achieved using a trial-and-error approach where further correspondences are added if the user is unsatisfactory with the deformation transfer results.

In this paper, we propose a novel approach to deformation transfer with automatic key point selection. Given a source shape and one or more deformed source shapes, as well as a target shape, deformation transfer produces the same number of deformed shapes with the same geometry as the target shape and the

deformation of the deformed source mesh transferred. Our major observation is that while it is difficult for an ordinary user with little experience to understand which correspondences are most effective, it is intuitive for users to specify the semantically meaningful point on the target shape that corresponds to a given point on the source shape. By producing a small set of essential key points, users are only required to specify their corresponding points on the target shape. Therefore, our technique can greatly reduce the time and expertise needed for deformation transfer. To the best of our knowledge, this is the first work that addresses the problem of automatic key point selection for deformation transfer. To achieve this, we adapt biharmonic weight shape deformation [5, 6] to solve the problem of deformation transfer, with improved clustering and an error cost suitable for deformation transfer. Extensive experiments show that our method outperforms state-of-the-art deformation transfer methods, and our automatically selected key points are more effective than those selected by ordinary users.

In the following sections, we first review the most related work to ours in Sec. 2. Algorithm details are then presented in Sec. 3, followed by experimental results and discussions in Sec. 4. Finally, we draw conclusions in Sec. 5.

2. Related Work

Shape deformation has received significant attention and many techniques have been developed to improve the representation capability to handle large-scale deformation, and utilize examples to produce better deformation results [7]. Please refer to [1, 2] for surveys of different deformation techniques. The recent work [8] develops an automatic method to deform meshes of arbitrary shapes to obtain their polycube form. The work [9] proposes a smooth, interpolating representation for shapes with spherical topology, and demonstrates its use for surface deformation. Many practical problems involve shape deformation. The work [10] studies stain formation and evolution on deforming cloths, and [11] exploits shape deformation for surgical simulation. In order to

improve realism, physics-based methods [12, 13] are also developed for shape deformation. In this work, we focus on transferring deformation from one shape to another, taking a simpler and more efficient data-driven geometry-based approach.

Global rigid transformation is not suitable when non-rigid deformation is involved. Instead, deforming the shape locally rigidly helps keep details while producing rich deformation results. The As-Rigid-As-Possible (ARAP) deformation energy is based on this idea, and has been widely used in geometric processing, such as shape manipulation [14, 15, 16, 17] and shape interpolation [18, 19]. Recent work [15] extends As-Rigid-As-Possible (ARAP) to anisotropic ARAP which is direction dependent, and can solve an important problem of flattening functional compression garments. Our work is based on [6], which is efficient and allows plausible deformation results to be produced, even with sparse key points.

We now focus on reviewing existing deformation transfer techniques which are most related to our work. In the pioneering work [3], the deformations of shapes are encoded using deformation gradients in local regions. With reliable correspondences between the source shape and the target shape, the deformation gradients are transferred to the target shape, which are then used to reconstruct the deformed target shape by solving Poisson equations. The method relies on accurate correspondences to work effectively, and requires quite a large number of correspondences due to the local nature of deformation gradients. In addition to transferring deformation, the deformation transfer results obtained using the above method may also contain geometric details from the source shape, which is undesirable and may produce unreasonable shapes. The work [20] improves over [3, 21] by adding an additional step of projecting the resulting shape to the manifold of plausible target shapes. The method however requires a set of target shapes that sufficiently covers the plausible deformation space, which is not always available.

The methods above can only handle triangle meshes. In order to deal with general shapes, cages (i.e. a set of polyhedra to enclose the shapes) are employed

to handle different shape representations such as triangle soups and tetrahedron meshes [22, 23]. These two works need extra effort to generate suitable cages which is not only time-consuming but also requires experience and expertise. Moreover, cages are sensitive to topological change and topological proximity of the models. For example, two points with a large geodesic distance can be close in Euclidean space, and so may be enclosed in the same cage and therefore deformed in the same way, which leads to unnatural deformation results. To deal with shapes with multiple components where each component is a manifold surface, an alternate solution is proposed using a graph structure to represent the general shapes for transferring the deformation gradients on the graph node [24]. This method requires the multi-component structure to be provided, and thus is not suitable for shapes without multiple components.

Instead of specifying correspondences on shapes, Baran et al. [4] propose a semantic deformation transfer method by exploiting the correlation between two shape sets (source and target). They assume that the source and target shape sets contain corresponding shapes with the same semantic meaning. Each deformed source shape is projected onto the source shape set, and the obtained combination weights are used along with the target shape set to produce the deformed target shape corresponding to the given source shape. The method achieves impressive results. However, it requires source and target shape sets with corresponding semantics as input which are only available in limited situations.

In this work, we address the problem of deformation transfer of meshes with the aim of significantly reducing user effort. Our method only requires one target shape as input, and does not require proxies such as cages. We generalize an efficient deformation method based on biharmonic weights to deformation transfer as it produces plausible results even with very few correspondences. We then develop an automatic key point selection algorithm such that the user is only required to specify points on the target shape corresponding to the key points that were produced automatically on the source shape, which is intuitive for ordinary users. Experimental results show that our method not only reduces

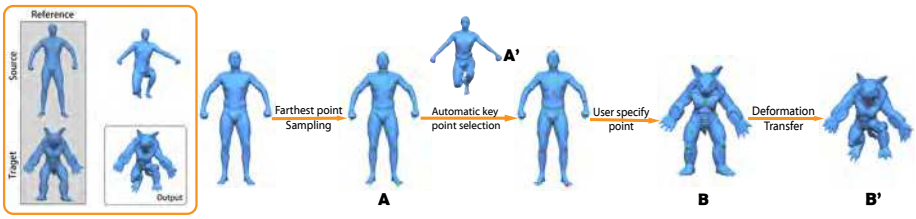


Figure 1: The pipeline of our algorithm.

user effort but also produces much better deformation transfer results than using correspondences specified by normal users, thanks to the effective choice of key points.

3. Our Algorithm

3.1. Algorithm Overview

The input to our algorithm is a source mesh $\mathbf{A}$ before deformation, a set of deformed source meshes $\mathcal{A}'$, and a target mesh $\mathbf{B}$, our deformation transfer algorithm produces a set of deformed target meshes $\mathcal{B}'$. For each mesh $\mathbf{A}' \in \mathcal{A}'$, a deformed target mesh $\mathbf{B}'$ is obtained by applying the deformation from $\mathbf{A}$ to $\mathbf{A}'$ to the target shape $\mathbf{B}$. Denote by $m = |\mathcal{A}'|$ the number of deformed source meshes. Note that in the simplest case, $\mathcal{A}'$ may only contain one deformed shape (i.e. $m = 1$). Note that $\mathbf{A}$ and meshes in $\mathcal{A}'$ share the same mesh connectivity, but the mesh topology of the source and target shapes can be different.

The pipeline of our algorithm is illustrated in Fig. 1. We first obtain a set of vertices on the source mesh as candidates for key points (denoted as C), by performing farthest point sampling [25, 26] to ensure candidate points provide sufficient coverage of the shape. Denote by $n_c = |C|$ the number of candidate points. Although depending on the random choice of the first candidate key point, farthest point sampling may generate different sets of candidate key points, our method produces very similar deformation transfer results even with substantially different candidate key points, as shown by the example in Fig. 2.

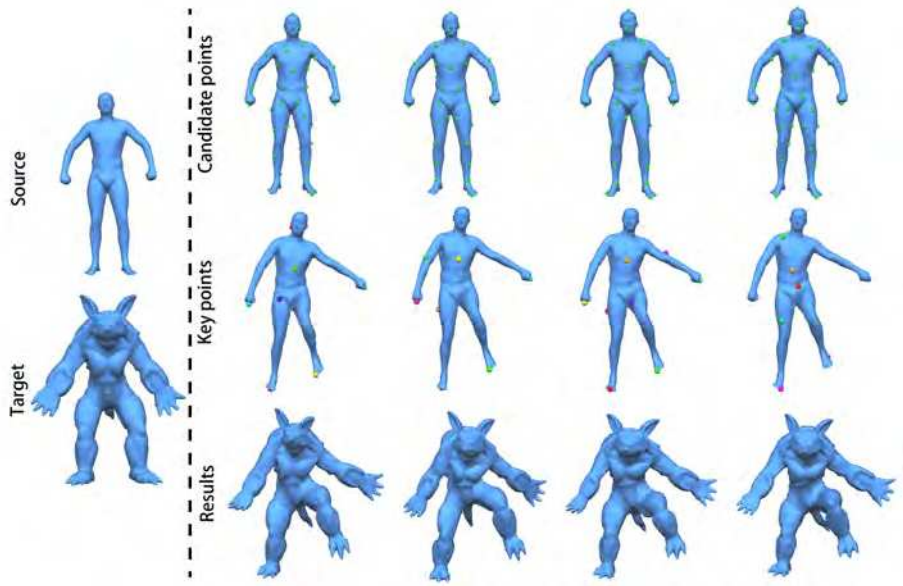


Figure 2: Comparison of deformation transfer results using different randomly initialized candidate key points. Left: the source mesh and the target mesh, right: deformation results. Every column shows a different set of randomly initialized candidate key points, our selected key points and corresponding deformation results. Similar deformation results are obtained even if the candidate key points are significantly different.

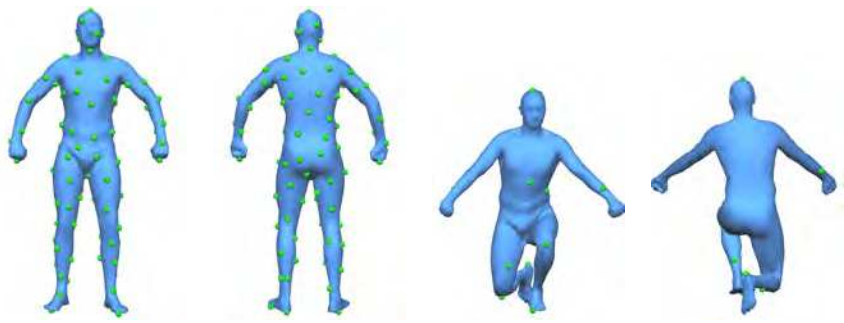
The key points S are then selected from the candidate set C. Denote by n_k the number of selected key points. Since the correspondences between the source and target meshes are not yet available and it is difficult to automatically judge the quality of deformed meshes, we take a practical approach aiming to find a key point set S that minimizes total deformation error from $\mathbf{A}$ to each mesh $\mathbf{A}' \in \mathcal{A}'$. A trivial solution would consider all the subsets of C as S and choose the best solution. This however involves $2^{n_c} - 1$ combinations and is prohibitively expensive. We propose to use a greedy approach, such that at each step, only one key point is optimized. Since initially only one or a few key points are selected and treated as handles to deform $\mathbf{A}$ towards models $\mathbf{A}' \in \mathcal{A}'$, deformation methods based on local deformation gradients (e.g. [27, 3, 21])

do not work well. We thus adapt the deformation method [6] with bounded biharmonic weights [5], by utilizing the deformed source shapes $\mathcal{A}'$ as constraints such that the deformed shapes are close to the desired shapes. Several energy functions used in shape deformation typically measure some forms of elastic shape distortion. As pointed out in the survey [28], using quadratic energies leads to linear optimization problems, which are robust and efficient to optimize, but result in linearization artifacts in the deformation results. So nonlinear energies [29, 30, 27, 31] are proposed to provide higher-quality deformation results, but they are generally slow to optimize. We use as-rigid-as-possible [14, 27, 32, 31] deformation along with clustering of the biharmonic weights to achieve high quality deformation while ensuring efficiency. Moreover, the deformations of neighboring vertices are highly correlated, so it is unnecessary to compute local rotation for each edge independently. Instead, by clustering local vertices into some clusters based on biharmonic weights, local regions are deformed consistently, which helps with both efficiency and deformation quality. We incrementally add or update key points until convergence. The user is then asked to specify points on $\mathbf{B}$ that correspond to the automatic selected key points S on $\mathbf{A}$. Finally, the resulting mesh $\mathbf{B}'$ with the deformation transferred is obtained using biharmonic weight-based mesh deformation using affine transformation of corresponding key points from the source mesh.

An example is shown in Fig. 3. We first apply farthest point sampling on the source mesh $\mathbf{A}$ and the candidates $n_c = 100$ are shown in Figs. 3 (a) and (b). They are well distributed, providing a sufficient set to choose key points from. The selected key points using our automatic algorithm are shown in Figs. 3 (c) and (d), and are effective in achieving the deformation from the original shape (a)(b) to the deformed shape (c)(d).

3.2. Shape Deformation using Biharmonic Weights

As a building block in our algorithm, we now introduce a shape deformation method using biharmonic weights. Since it is used for deforming both source shapes (for optimization of key points) and target shapes (for deformation trans-



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

Figure 3: An example demonstrating candidate points and selected key points for deforming from (a)(b) to (c)(d). (a) and (b) are the candidates obtained using farthest point sampling (front and back views, with $n_c = 100$ candidate points), (c) and (d) are the key points automatically selected by our algorithm (front and back views, $n_k = 12$ key points).

fer), we describe the algorithm using a generic set of symbols. Given an input mesh before deformation $\mathcal{P}$, let $\mathcal{Q}$ be the deformed mesh. $\mathbf{p}_i \in \mathcal{P}$ and $\mathbf{q}_i \in \mathcal{Q}$ are the positions of the i^{th} vertex of the mesh $\mathcal{P}$ and $\mathcal{Q}$ respectively. Both meshes have the same connectivity. Denote by $n_p = |\mathcal{P}|$ the number of vertices of both meshes. For the purpose of deformation, assuming $\mathcal{H}$ is the set of handle vertices, and $n_h = |\mathcal{H}|$ is the number of handles. For each handle $h_k \in \mathcal{H}$, it is associated with an affine transformation $\mathbf{T}_k \in \mathbb{R}^{3\times 4}$. For simplicity, these affine transformations are packed into a matrix $\mathbf{T}$ of size $12n_h \times 1$ (column vector) by stacking each affine transformation as a 12-dimensional column vector. When applying the deformation method to source meshes, the deformed mesh is known, and denoted as $\mathcal{Q}'$ with $\mathbf{q}'_i$ representing the i^{th} vertex of the known deformed mesh.

Similar to [6], the position of vertices on the deformed mesh $\mathcal{Q}$ can be computed by applying affine transformations $\mathbf{T}$ with linear blend skinning. Denote by $\mathbf{W} \in \mathbb{R}^{n_p \times n_h}$ the skinning weights, where $\mathbf{W}_{ph}$ is the influence that the h^{th} handle has on the p^{th} vertex. The skinning weights can be defined in many ways, including manually specified by artists. In our implementation, we use

the bounded-biharmonic weight [5], which is known to be suitable for deformation. Following [5], we compute the bounded-biharmonic weights with the optimization below:

$$\begin{aligned} &\arg\min_{w_k} \sum_{k=1}^{n_h} \frac{1}{2} \int_{\mathbf{p}\in\mathcal{P}} \|\Delta w_k\|^2 d\mathbf{p} \\ &\text{subject to }: w_k(\mathbf{p}_j) = \delta_{jk} \\ &\qquad \sum_{k=1}^{n_h} w_k(\mathbf{p}) = 1 \qquad\qquad \forall \mathbf{p} \in \mathcal{P} \\ &\qquad 0 \le w_k(\mathbf{p}) \le 1, k = 1, \ldots, n_h \quad \forall \mathbf{p} \in \mathcal{P} \end{aligned} \tag{1}$$

where $\mathbf{W}_{jk} = w_k(\mathbf{p}_j)$ is the skinning weight of the j^{th} vertex of the mesh w.r.t. the k^{th} handle vertex of the mesh, w_k is a function over the space in which the mesh is embedded, and δ_{jk} is Kronecker's delta ($\delta_{jk} = 1$ if $j = k$ and 0 otherwise). This is consistent with [5]; please refer to the paper for more details.

Using linear blend skinning, the i^{th} vertex position $\mathbf{q}_i$ of the deformed mesh $\mathcal{Q}$ is given as follows:

$$\mathbf{q}_i = \sum_{k=1}^{n_h} \mathbf{W}_{ik} \mathbf{T}_k \begin{bmatrix} \mathbf{p}_i \\ 1 \end{bmatrix} \tag{2}$$

To measure the quality of deformation, following [6], we use an as-rigid-as-possible (ARAP) energy [27] E_{arap} with deformed positions obtained using Eqn. 2. To better preserve (near) piecewise rigidity and avoid over-fitting, the shape is partitioned into a set of regions $\mathcal{G} = \{\mathcal{G}_g\}$, $g = 1, 2, \ldots, |\mathcal{G}|$ and $|\mathcal{G}|$ is the number of regions (treated as edge groups). The details of the partitioning algorithm will be introduced in Sec. 3.3. A local rotation matrix $\mathbf{R}_g$ is assigned for each region $\mathcal{G}_g$. The energy can be written as:

$$E_{arap} = \sum_{g} \sum_{(i,j)\in\mathcal{G}_g} \tilde{w}_{ij} \left\| (\mathbf{q}_i - \mathbf{q}_j) - \mathbf{R}_g(\mathbf{p}_i - \mathbf{p}_j) \right\|_2^2 \tag{3}$$

where $\tilde{w}_{ij}$ is a cotangent weight [33] which is useful for meshes with irregular triangulation, and $\mathbf{R}_g \in \mathrm{SO}(3)$ is the rotation of the edge group g.

For source meshes, since the deformed mesh $\mathcal{Q}'$ is known, we further introduce another energy term that measures the difference of the mesh obtained by the deformation and the known deformed mesh. This penalizes meshes that deviate too much from the known results.

$$E_{diff} = \sum_{i=1}^{n_p} \left\| \mathbf{q}_i - \mathbf{q}_i^{'} \right\|_2^2 \tag{4}$$

The overall energy is obtained by a linear combination of both energy terms:

$$E = \lambda E_{arap} + E_{diff}, \tag{5}$$

where λ is a weight to balance the two terms. We set $\lambda = 0.5$ in our experiments. The energy aims to make the resulting mesh as close as possible to the known deformed mesh, while keeping the local shapes by reducing the ARAP energy. As we will show later, this helps to identify better transformations to better reproduce the deformed mesh, and thus helps improve deformation transfer results. The unknowns in this function include affine transformation $\mathbf{T}_k$ of each handle h_k, and rotation matrix $\mathbf{R}_g$ for each edge group g of the mesh. Note that the deformed mesh $\mathcal{Q}$ is determined once the affine transformations $\mathbf{T}$ are given. We alternately optimize $\mathbf{T}$ and $\mathbf{R}$; see Sec. 3.4 for details of the optimization.

3.3. Clustering with Skinning Weights and Rotation

As suggested by [6], we can obtain a segmentation of the mesh by using k-means clustering on the skinning weight matrix $\mathbf{W}$, as it shows how different handles contribute to the deformation of each vertex. The clustering of shapes is derived from the result of key point selection. The number of clusters is the same as the number of key points, i.e. we set the number of clusters to n_h. The clustering helps identify regions of the mesh with consistent deformation transformation. For deformation transfer, we also have a set of deformed source meshes $\mathcal{A}'$. It is therefore possible to exploit the local rotations of these meshes, to help identify regions with consistent deformation. This provides useful additional information not available from $\mathbf{W}$.

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

Figure 4: K-means clustering on the biharmonic weight matrix $\mathbf{W}$ and rotation-augmented weight matrix $\mathbf{W}'$. The number of clusters n_c is set to 13. (a) shows the handle points selected by the user. (b) is the result of k-means clustering on $\mathbf{W}$, (c) is the result of k-means clustering only based on **logr** and **s** without $\mathbf{W}$ using the 55^{th} model in the SCAPE dataset as the deformed source shape, (d) is the result of k-means clustering on $\mathbf{W}'$ including $\mathbf{W}$, **logr** and **s** using the 55^{th} model in the SCAPE dataset, (e) is the result of k-means clustering on $\mathbf{W}'$ of all the 71 models in the SCAPE dataset [34].

To achieve this, for each mesh $\mathbf{A}' \in \mathcal{A}'$, we first compute the local deformation gradient $\mathbf{D}_i$ for the i^{th} vertex of $\mathbf{A}'$, which is calculated by minimizing the following energy:

$$E(\mathbf{D}_i) = \sum_{j \in N_i} \tilde{w}_{ij} \left\| \mathbf{e}_{ij}^q - \mathbf{D}_i \mathbf{e}_{ij}^p \right\|^2 \tag{6}$$

where N_i is the 1-ring neighbors of the vertex i, $\mathbf{e}_{ij}^q := \mathbf{q}_i - \mathbf{q}_j$ and $\mathbf{e}_{ij}^p := \mathbf{p}_i - \mathbf{p}_j$. The deformation gradient $\mathbf{D}_i$ can be decomposed into the product of a rotation matrix and a scale/shear matrix by polar decomposition [35]:

$$\mathbf{D}_i = \mathbf{U}_i \mathbf{N}_i \tag{7}$$

where $\mathbf{U}_i$ is a 3×3 rotation matrix and $\mathbf{N}_i$ is a 3×3 symmetric matrix that represents the scaling/shear on the three orthogonal axes. Then the rotation matrix can be mapped to space $so(3)$ by the matrix logarithm operation: $\bar{\mathbf{U}}_i = \log \mathbf{U}$, which is known to make the space more linear. Because the matrix $\bar{\mathbf{U}}$ is a skew-symmetric matrix, we can rewrite the $\bar{\mathbf{U}}$ in the space $so(3)$ that consists

of three orthogonal basis vectors [36]:

$$\bar{\mathbf{U}} = u_i^{(1)}\mathbf{e}_1 + u_i^{(2)}\mathbf{e}_2 + u_i^{(3)}\mathbf{e}_3 \tag{8}$$

where

$$\mathbf{e}_1 = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{e}_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \mathbf{e}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \tag{9}$$

and $u_i^{(1)}, u_i^{(2)}$, and $u_i^{(3)} \in \mathbb{R}$. We then obtain a vector $\mathbf{u}_i$ for each vertex:

$$\mathbf{u}_i = \left(u_i^{(1)}, u_i^{(2)}, u_i^{(3)}\right) \tag{10}$$

Similarly, the scaling/shear matrix can be rewritten as a long vector

$$\mathbf{s}_i = \left(n_i^{(1)}, n_i^{(2)}, \ldots, n_i^{(9)}\right) \tag{11}$$

The rotation logarithm matrix **logr** for a deformed mesh is defined as:

$$\mathbf{logr} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \ldots & \mathbf{u}_{n_p} \end{bmatrix}^T \tag{12}$$

and the scaling/shear matrix **s** for a deformed mesh is defined as:

$$\mathbf{s} = \begin{bmatrix} \mathbf{s}_1 & \mathbf{s}_2 & \ldots & \mathbf{s}_{n_p} \end{bmatrix}^T \tag{13}$$

where n_p is the number of vertex. We collect all these matrices corresponding to meshes in $\mathcal{A}'$ as

$$\widetilde{\mathbf{logr}} = [\mathbf{logr}_1, \mathbf{logr}_2, \ldots, \mathbf{logr}_m], \quad \widetilde{\mathbf{s}} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_m] \tag{14}$$

where $\mathbf{logr}_j$ and $\mathbf{s}_j$ are the **logr** and **s** matrices for the j^{th} model of $\mathcal{A}'$. Finally, we augment $\mathbf{W}$ as follows:

$$\mathbf{W}' = \begin{bmatrix} \mathbf{W} & \frac{\gamma_{\mathbf{logr}}}{\sqrt{m}}\widetilde{\mathbf{logr}} & \frac{\gamma_{\mathbf{s}}}{\sqrt{m}}\widetilde{\mathbf{s}} \end{bmatrix}, \tag{15}$$

$\sqrt{m}$ is used for normalization since the k-means clustering uses squared Euclidean distance.

Fig. 4 shows a comparison of clustering results using $\mathbf{W}$ and $\mathbf{W}'$ on the SCAPE dataset [34]. It can be seen that the segmentation obtained using $\mathbf{W}$

(Fig. 4b) does not always represent the correct rigid components and the boundaries of segments can also be inaccurate. When using the rotation/scaling alone without $\mathbf{W}$, the segmentation is quite noisy (Fig. 4c). By using our augmented matrix $\mathbf{W}'$ combining both biharmonic weights $\mathbf{W}$ and rotation/scaling (**logr** and **s**), the result is significantly better even with only one deformed example (Fig. 4d), and further improved with the whole dataset (Fig. 4e). $\gamma_{\mathbf{logr}}$ and $\gamma_{\mathbf{s}}$ are the adjustable parameters, and by default we choose $\gamma_{\mathbf{logr}} = 1, \gamma_{\mathbf{s}} = 0.1$.

3.4. Algorithmic Solution of Our Deformation Method

Similar to [6, 27], the optimization of our deformation method can also be solved by alternating two steps, namely the Global Step and the Local Step.

In the Global Step, we fix $\mathbf{R}_g$ for each edge group, and optimize the energy E to obtain deformed positions $\mathbf{q}_i$. For the as-rigid-as-possible (ARAP) energy, we set $\frac{\partial E_{arap}}{\partial \mathbf{q}_i} = \mathbf{0}$, and Eqn. 3 can be rewritten as a system of linear equations

$$\begin{aligned} &\sum_{g} \sum_{(i,j)\in\mathcal{G}_g} \tilde{w}_{ij}(\mathbf{q}_i - \mathbf{q}_j) \\ =&\sum_{g} \sum_{(i,j)\in\mathcal{G}_g} \tilde{w}_{ij}\mathbf{R}_g(\mathbf{p}_i - \mathbf{p}_j) \end{aligned} \tag{16}$$

Eqn. 16 can be written in a matrix form as:

$$\mathbf{L}\mathbf{q} = \mathbf{b} \tag{17}$$

where $\mathbf{L}$ is the Laplace matrix, $\mathbf{q} = [\mathbf{q}_1, \ldots, \mathbf{q}_{n_p}]^T$ is the deformed positions to be determined, and $\mathbf{b}$ is the right hand side coefficients.

To minimize E, we add the terms related to E_{diff} to Eqn. 17 and obtain the following linear system:

$$\begin{bmatrix} \lambda\mathbf{L} \\ \mathbf{I} \end{bmatrix} \mathbf{q} = \begin{bmatrix} \lambda\mathbf{b} \\ \mathbf{q}' \end{bmatrix} \tag{18}$$

where $\mathbf{I}$ is the n-dimensional identity matrix, and $\mathbf{q}'$ is the vertex position of the known deformed source model.

Next, we put Eqn. 2 into Eqn. 18, and obtain the following equations:

$$\begin{bmatrix} \lambda\mathbf{L} \\ \mathbf{I} \end{bmatrix} \begin{bmatrix} \sum_k \mathbf{W}_{1k}\mathbf{T}_k \begin{pmatrix} \mathbf{p}_1 \\ 1 \end{pmatrix} \\ \sum_k \mathbf{W}_{2k}\mathbf{T}_k \begin{pmatrix} \mathbf{p}_2 \\ 1 \end{pmatrix} \\ \vdots \\ \sum_k \mathbf{W}_{n_p k}\mathbf{T}_k \begin{pmatrix} \mathbf{p}_{n_p} \\ 1 \end{pmatrix} \end{bmatrix} = \begin{bmatrix} \lambda\mathbf{b} \\ \mathbf{q}' \end{bmatrix} \tag{19}$$

Eqn. 19 can be further represented as:

$$\mathbf{SMT} = \mathbf{b}' \tag{20}$$

where $\mathbf{S} = \begin{bmatrix} \lambda\mathbf{L} \\ \mathbf{I} \end{bmatrix}$, $\mathbf{M}$ is a $(3n_p) \times (12n_h)$ sparse matrix, $\mathbf{b}'$ is the right hand side of Eqn. 18, and $\mathbf{T} \in \mathbb{R}^{12n_h \times 1}$ (a column vector) contains all the affine transformations. We can pre-compute $\mathbf{SM}$ and obtain its LU decomposition to accelerate solving Eqn. 20, and obtain $\mathbf{T}$ needed for deformation transfer.

The second step is the Local step. Given $\mathbf{T}$, we can obtain the vertex position of the deformed mesh $\mathbf{q}$ using Eqn. 2. We then find the optimal $\mathbf{R}_g$ for each edge group g. Let us denote the edge vector $\mathbf{e}_{ij}^q := \mathbf{q}_i - \mathbf{q}_j$ and $\mathbf{e}_{ij}^p := \mathbf{p}_i - \mathbf{p}_j$. Minimizing Eqn. 5 can be solved independently. For edge group g, this is achieved by maximizing the following:

$$\arg\max_{\mathbf{R}_g} Tr\left(\mathbf{R}_g \sum_{(i,j)\in\mathcal{G}_g} \tilde{w}_{ij}\mathbf{e}_{ij}^p \mathbf{e}_{ij}^{q^T}\right) \tag{21}$$

where $Tr(\cdot)$ is the matrix trace. According to [27], the above optimization has a closed form solution and the optimal $\mathbf{R}_g$ can be obtained using singular value decomposition (SVD). Let us denote $\hat{\mathbf{S}}_g = \sum_{(i,j)\in\mathcal{G}_g} \tilde{w}_{ij}\mathbf{e}_{ij}^p \mathbf{e}_{ij}^{q^T}$. Then, using SVD, $\hat{\mathbf{S}}_g = \hat{\mathbf{U}}_g\hat{\mathbf{\Sigma}}_g\hat{\mathbf{V}}_g$. $\mathbf{R}_g$ can be obtained as $\hat{\mathbf{V}}_g\hat{\mathbf{U}}_g^T$. If the resulting $\mathbf{R}_g$ does not satisfy $\det\mathbf{R}_g > 0$, we negate it to ensure the obtained matrix is a rotation matrix (rather than a mirrored matrix). We alternate the Global Step and the Local Step until convergence (i.e. the energy stays stable).

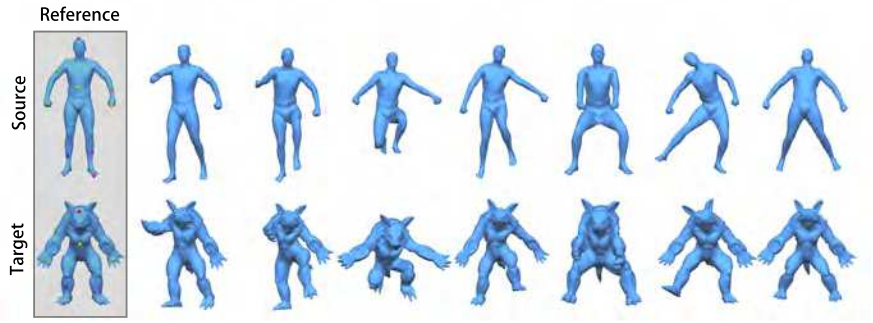


Figure 5: Results of transferring the deformation on the source mesh (a human from the SCAPE dataset) onto a different target mesh (the Armadillo model) using our method with automatic key point selection. Correspondences are highlighted using colored balls where the same color indicates corresponding points. The first column contains the source and target meshes without deformation. The first row shows the source meshes and the second row gives the output meshes. The deformations of input meshes are reproduced successfully on the target mesh, even with substantial geometric difference and large deformations.

3.5. Automatic Key Point Selection

Automatic key point selection aims to find a subset $S \subset C$ from the candidate set C. To make the problem tractable, we use a greedy approach. The algorithm works in two stages. In the first stage, we incrementally add new candidate key point to S, and in the second stage, we try to improve existing key points in S.

In the first stage, we start by setting $S = \{c_1\}$. Since we will later update key points in the set, the choice of the first key point does not usually affect the results. We then iteratively add a new key point c_t to S, which is the one that leads to the minimum energy:

$$\hat{E} = \frac{1}{mn_p} \min_{c_t \in C-S} \sum_{\mathbf{A}' \in \mathcal{A}'} \|D_{S\cup\{c_t\}}(\mathbf{A}) - \mathbf{A}'\|_F, \tag{22}$$

where $D_S(\cdot)$ is an operator that produces the deformed mesh with S as key points, n_p is the number of vertices, and m is the number of models. The process repeats until the resulting energy $\hat{E}$ is sufficiently small (under a threshold $\varepsilon = 0.03$, where the models are scaled consistently to fit into a unit sphere). The normalization makes the same error threshold applicable to a wide range

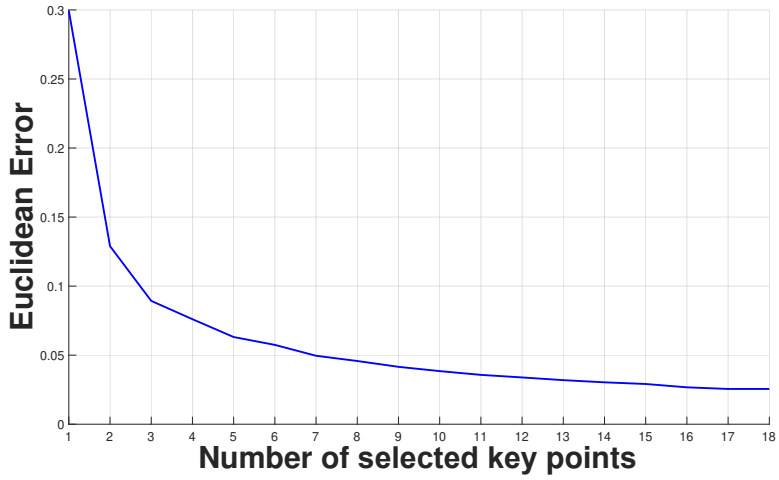


Figure 6: The Euclidean distance between $\mathbf{A}'$ and the deformed $\mathbf{A}$ using the example in Fig. 5. The Euclidean distance decreases quickly and converges with a small number of key points.

of datasets.

In the second stage, we try to replace each selected key point in turn. For key point $c_t \in S$, we aim to find the best replacement while keeping other key points unchanged:

$$c_t^* = \underset{c_j \in C-S \cup \{c_t\}}{\arg\min} \sum_{\mathbf{A}' \in \mathcal{A}'} \|D_{S-\{c_t\}\cup\{c_j\}}(\mathbf{A}) - \mathbf{A}'\|_F. \tag{23}$$

We then replace c_t with c_t^*. This process guarantees the error is non-increasing, as if no better alternative exists, c_t will remain unchanged. This repeats until no further improvement can be found.

The pseudocode of the algorithm is summarized in Algorithm 1.

3.6. Deformation Transfer

After automatic key point selection, we use the method [6] to obtain the transformation $\mathbf{T}$ associated with each key point to deform the source mesh $\mathbf{A}$ to its deformed shape $\mathbf{A}^{'}$. Then we ask users to select key points on the target reference mesh $\mathbf{B}$ corresponding to the automatically selected key points on $\mathbf{A}$.

Algorithm 1 Algorithm for Automatic Key Point Selection

Input: Source mesh $\mathbf{A}$, source deformed mesh set $\mathcal{A}'$, the set of candidate points from farthest point sampling $C = \{c_1, c_2, \ldots, c_{n_c}\}$, where n_c is the number of candidate points. $n_c = 100$ is used in our experiments. $n_k \leq n_c$ is the number of selected key points. ε is the threshold for termination of adding key points.

Output: The set of selected key points S, the affine matrix $\mathbf{T}$.

Initialize $S = \varnothing$
Initialize $error = \infty$
Add c_1 into S, $C = C - \{c_1\}$
while $error > \varepsilon$ do ▷ first optimization
    for $c_i \in C$ do
        $sum_i = 0$
        for $\mathbf{A}_j^{'} \in \mathcal{A}'$ do
            Let the desired deformed mesh $\mathcal{Q}_j' = \mathbf{A}_j^{'}$ and use $S \cup \{c_i\}$ as handles
            Solve Eqn. 5 to obtain deformed vertex positions $\mathcal{Q}_j$
            $err_j = \frac{1}{mn_p} \left\| \mathcal{Q}_j - \mathcal{Q}_j^{'} \right\|_F$
            $sum_i = sum_i + err_j$
        end for
    end for
    Let $t = \arg\min_i sum_i$ be the index with the minimum error. Add c_t to S, and remove c_t from C.
    Set $error = sum_t$.
end while
Get the key point set S, and $S \cup C = \{c_1, c_2, \ldots, c_{n_c}\}$
repeat ▷ second optimization
    $\forall c_i \in S$, move c_i from S into C
    Find the optimal key point c_t in C, move c_t from C into S
until the set S is not changed
Return S and the corresponding $\mathbf{T}$.

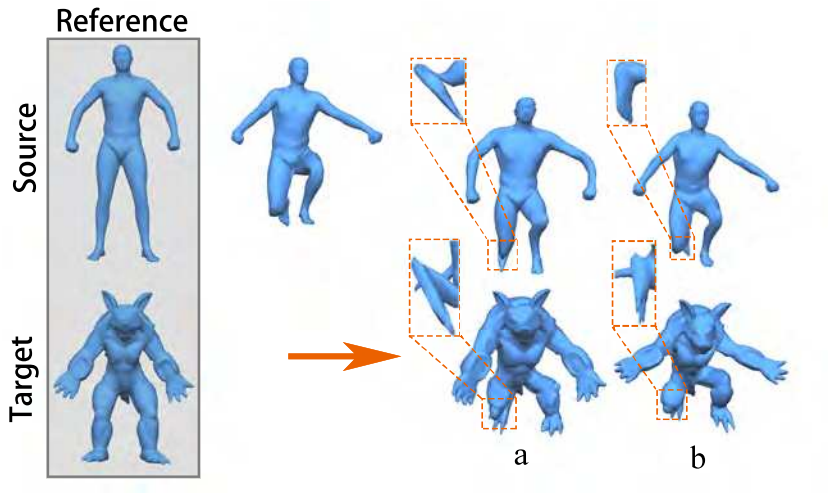


Figure 7: Comparison of deformation results (top row) and deformation transfer results (bottom row) without (a) and with (b) the E_{diff} term.

Once this is done, we directly apply the transformation matrix $\mathbf{T}$ of each key point from the source reference mesh $\mathbf{A}$ to the corresponding point of the target reference mesh $\mathbf{B}$, and use the method [6] again to obtain the deformed mesh $\mathbf{B}'$ by Eq. 2.

4. Results and Evaluation

Our experiments were carried out on a computer with an Intel i7-6850K CPU and 16GB RAM. The algorithm complexity w.r.t. the number of candidate sample points n_c is $O(n_c^2)$. Since the calculation of errors with a different added key point can be performed independently, we parallelize the algorithm using OpenMP. The running times for key point selection, biharmonic weight calculation and deformation transfer for different examples in the paper are reported in Table 1. The key point selection process takes between a few minutes to about half an hour, whereas the deformation transfer is under a minute. Note that key point selection can be considered as an offline preprocessing step so the running time is acceptable.

Figure	Source (#V/#F)	Target (#V/#F)	Key Point Selection (hours)	SWT/h (s)	LBS Time (ms)
Fig. 5	2161/4318	4502/9000	0.2831	1.342	0.58
Fig. 9	2752/5500	6890/13776	0.1452	4.164	1.3
Fig. 13	2161/4318	4526/9028	0.2085	1.664	0.47
Fig. 10	2502/5000	5012/10000	0.6938	1.888	0.78
Fig. 15	1127/2129	5050/9999	0.084	1.856	0.31
Fig. 11	2502/5000	5002/10000	0.2957	2.138	1.1
Fig. 12	1856/3708	2161/4318	0.0534	0.448	0.098

Table 1: Statistics of running times of automatic key point selection and deformation transfer. SWT/h is the time of calculating skinning weights per handle. In the last column of the table, the LBS Time is the time for linear blend skinning, i.e. calculating Eqn. 2.

We used various datasets to compare with the existing research [22, 3]. These various datasets come from [3] (Horse, Flamingo), SCAPE [34], TOSCA [25] (Dog, Gorilla, Micheal), MPI DYNA [37] (Fig. 14), MPI FAUST [38] (Fig. 9), FaceWareHouse [39] (Fig. 15), Cactus and Armadillo. When compared with [22], we used the released code. In this section, we will show various examples to demonstrate the performance of our method and compare it with the existing state-of-the-art methods.

Fig. 5 shows the results of transferring human deformation from the SCAPE dataset to the Armadillo model. It can be seen that the human and armadillo models differ significantly in geometry, and our method with automatic key point selection effectively produces high-quality deformation transfer results with a very sparse set of correspondences (highlighted as colored balls). We further show the Euclidean error with an increasing number of key points selected in Fig. 6. It shows that the energy decreases quickly and converges with a small number of key points. To show the effect of incorporating E_{diff} for deformation transfer, we compare the results (a) without and (b) with this term in Fig. 7. The top row shows the deformation of the source model. The E_{diff} term helps

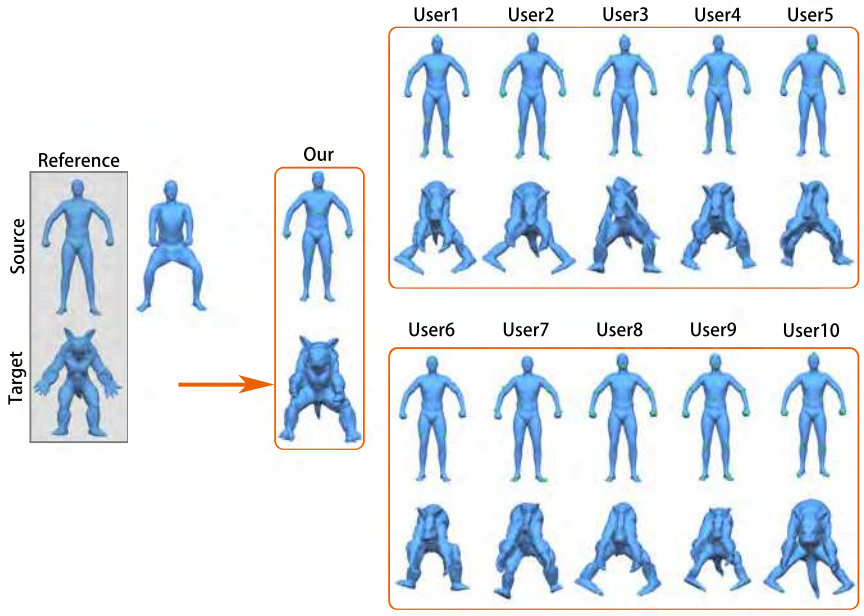


Figure 8: Comparison deformation transfer results obtained with automatic key point selection and user manual selection.

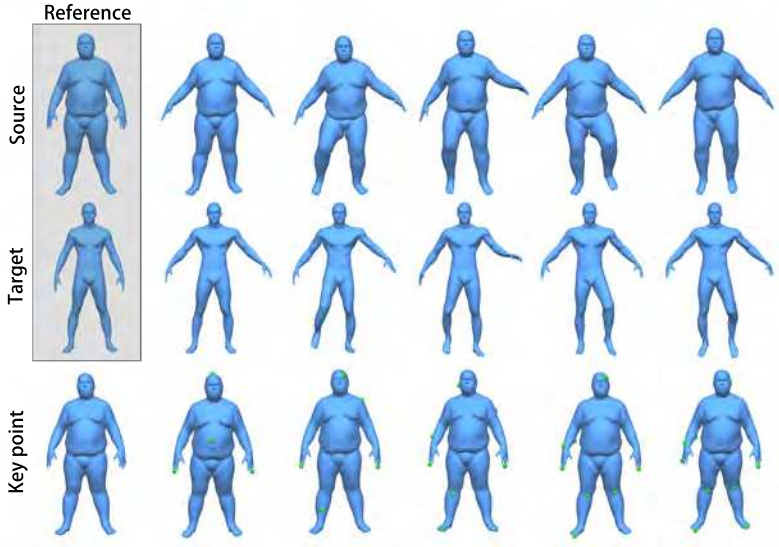


Figure 9: Deformation transfer results on sequences of the MPI DYNA dataset. From left to right, we incrementally add new shapes to $\mathcal{A}'$. The bottom row shows the key points that are selected by our algorithm with increasingly large $\mathcal{A}'$.

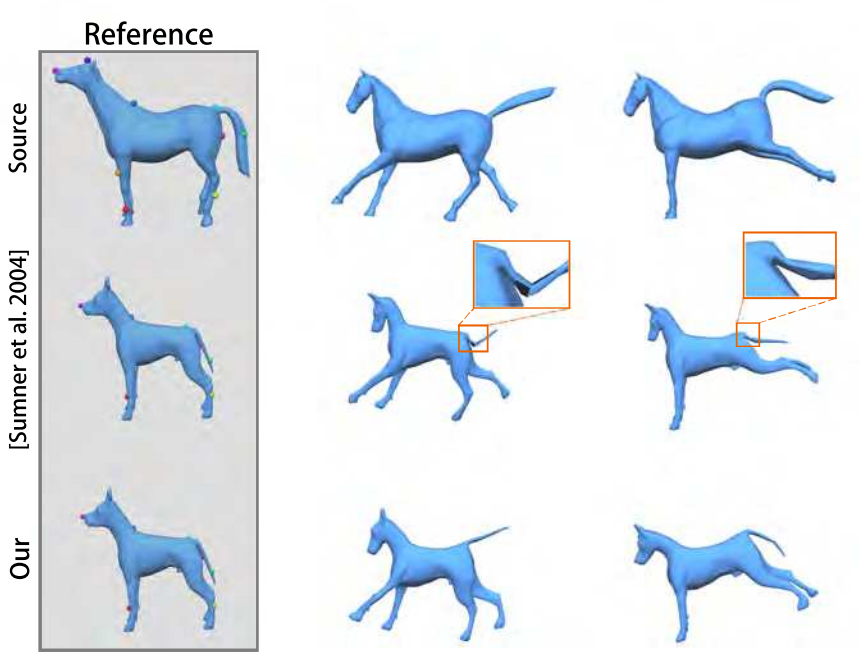


Figure 10: Results of transferring the deformation of a horse to a dog. The first column shows the source and target meshes with correspondences highlighted. Top row: source meshes, second row: the results of [3], bottom row: our results.

to make the deformation result much closer to the given deformed source shape $\mathbf{A}'$. As a result, this also helps improve the deformation transfer result (bottom row).

To evaluate the effectiveness of key point selection, we performed a user study. 10 participants were involved in the user study where they were asked to choose n_k correspondences manually. Results for the human to armadillo transfer example are shown in Fig. 8. The deformation transfer result using our deformation transfer framework but with manual correspondences performs significantly worse than the result with our automatically selected key points, with obvious artifacts, including distortions and dissimilarity of poses. Our automatic key point selection not only reduces user effort but produces much

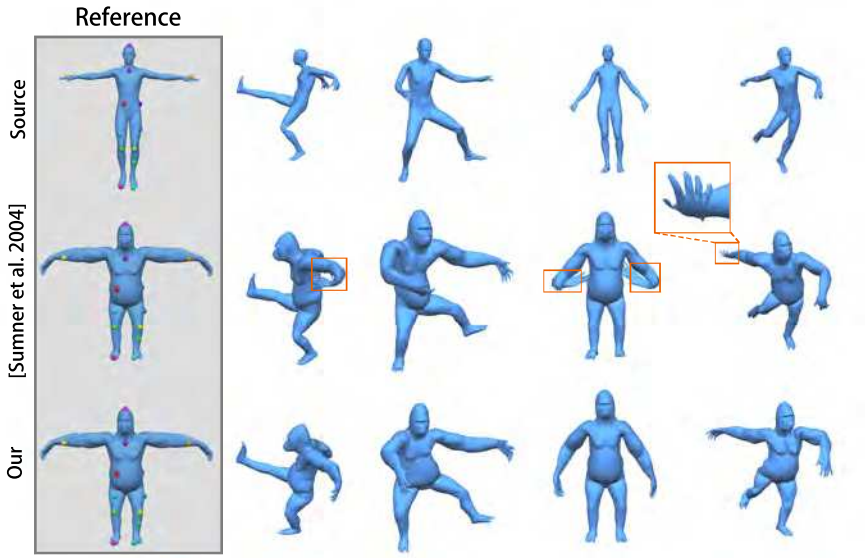


Figure 11: Results of transferring the deformation of a person to a gorilla. The first column shows the source and target meshes with correspondences highlighted. Top row: source meshes, second row: the results of [3], bottom row: our results.

more realistic deformation transfer results.

We further evaluate how our key point selection copes with a larger set of deformed source shapes $\mathcal{A}'$. Fig. 9 shows an example based on the MPI DYNA dataset. The results from left to right show key points selected with more shapes added to $\mathcal{A}'$. It can be seen that the selected key points are updated to reflect the needs of newly added shapes.

We also compare our deformation transfer method with state-of-the-art deformation transfer methods [22, 3] using a variety of examples (Figs. 10-13). These examples are challenging as the source and target shapes differ significantly (e.g. a cactus vs. a person in Fig. 12, and a person vs. a flamingo in Fig. 13) and contain large deformations. Our method produces plausible deformation transfer results which are artifact-free and semantically correct. Alternative methods [22, 3] can create distorted output due to too few correspondences, such as dissimilar deformations from the source deformation and im-

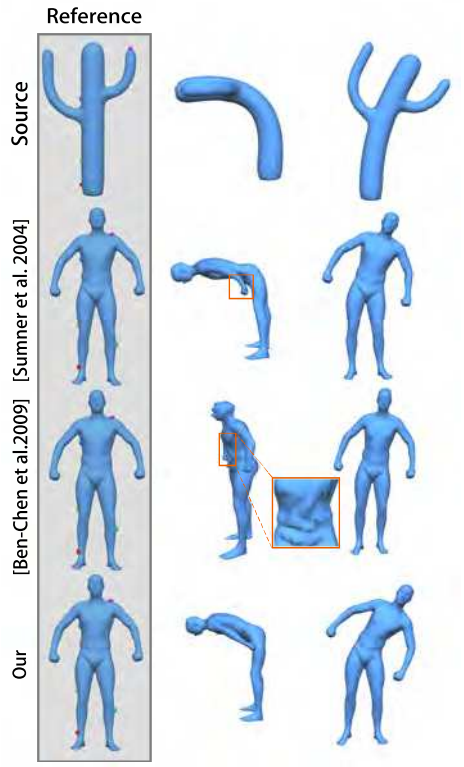


Figure 12: Results of transferring the deformation of a cactus to a person. The first column shows the source and target meshes. Top row: source meshes, second row: the results of [3], third row: the results of [22], bottom row: our results.

plausible shapes (e.g. wrongly bent legs of the flamingo). Since the method [22] uses cages, additional effort is needed to create such cages. For some examples, cages may include additional parts of the mesh, causing poor deformation results. Artifacts of these methods are highlighted using red rectangles.

It is generally difficult to provide a quantitative evaluation for deformation transfer methods. We use the MPI FAUST dataset which contains human bodies of different shapes with the same set of poses (see Fig. 14). We can therefore use it for computing a numerical measure taking the target shape with desired pose as the ground truth. We use both our automatically selected key points and the manually specified ones (the best result out of the 10 participants) and compare

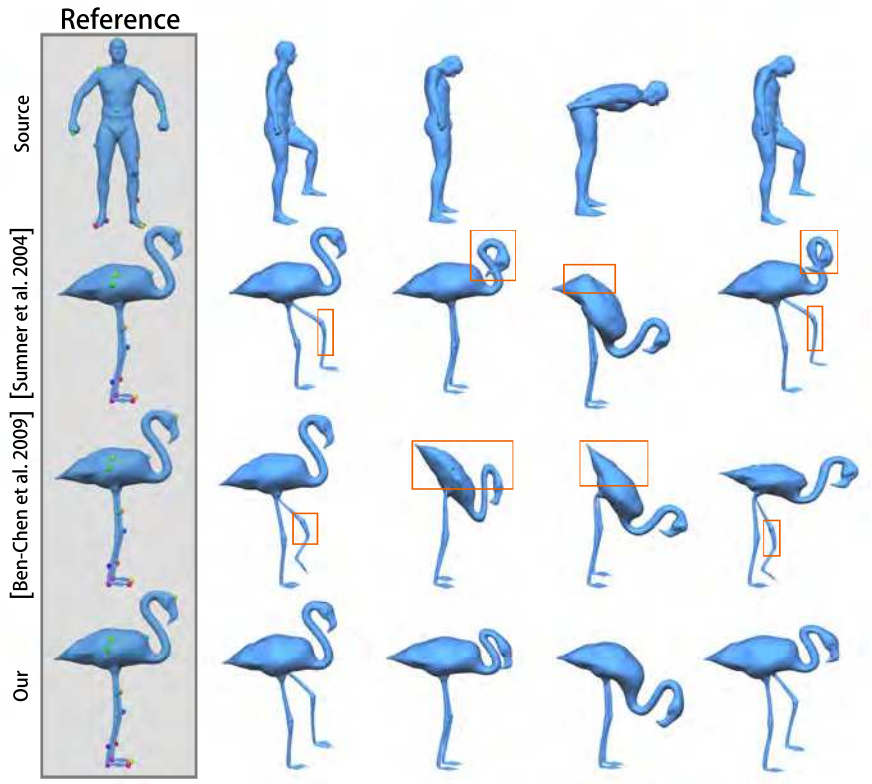


Figure 13: Results of transferring the deformation of a person from the SCAPE dataset to a flamingo. The first column shows the source and target meshes. Top row: source meshes, second row: the results of [3], third row: the results of [22], bottom row: our results.

deformation transfer results with our method and alternative methods [22, 3]. We measure the average Euclidean distance between corresponding vertices of the deformation transfer results and the ground truth. We show the proportion of correspondences (y-axis) within an error bound (x-axis) of different results. Our method is consistently better than the alternative methods. Moreover, for our method, our automatically selected key points outperform user specified key points.

We also show a challenging example of transferring human facial expressions to a dog (see Fig. 15). Our method is able to produce natural deformation

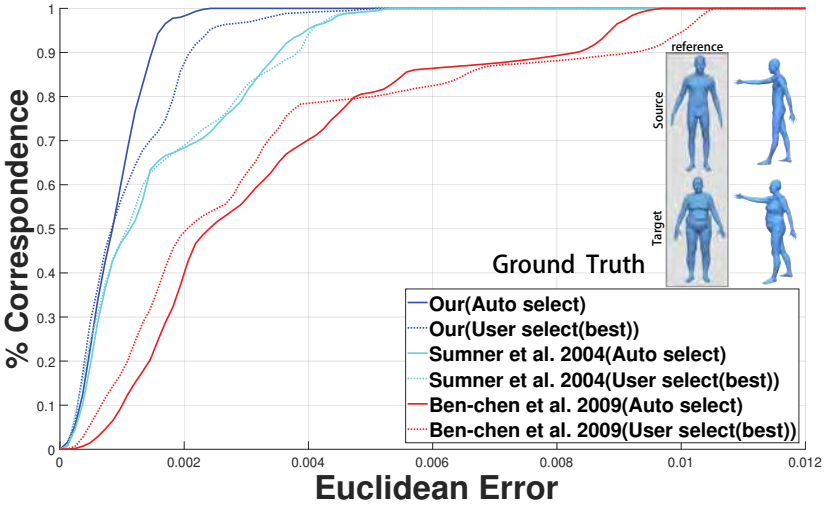


Figure 14: Comparison with methods [22, 3] on the MPI FAUST dataset. We show the proportion of correspondences (y-axis) within an error bound (x-axis) with results generated by different deformation transfer methods, as well as automatic and manually selected key points.

results even with a large difference of shapes.

5. Conclusions

In this paper, we adapt skinning with biharmonic weights to deformation transfer, and provide an automatic method to select effective key points. According to the amount of deformation and the level of deformation details, our method adaptively selects a suitable number of key points, as well as their positions, such that good transfer results are obtained. Therefore, if the source deformed mesh $\mathbf{A}'$ has more deformation details, more key points will be selected. Nevertheless, the number of key points required is still less than traditional methods [3]. The aim of our method is to obtain effective deformation transfer with as few key points as possible. We exploit deformed source meshes to provide better segmentation and add an additional constraint to ensure the deformed shape is close to the given deformed source meshes. Our deformation

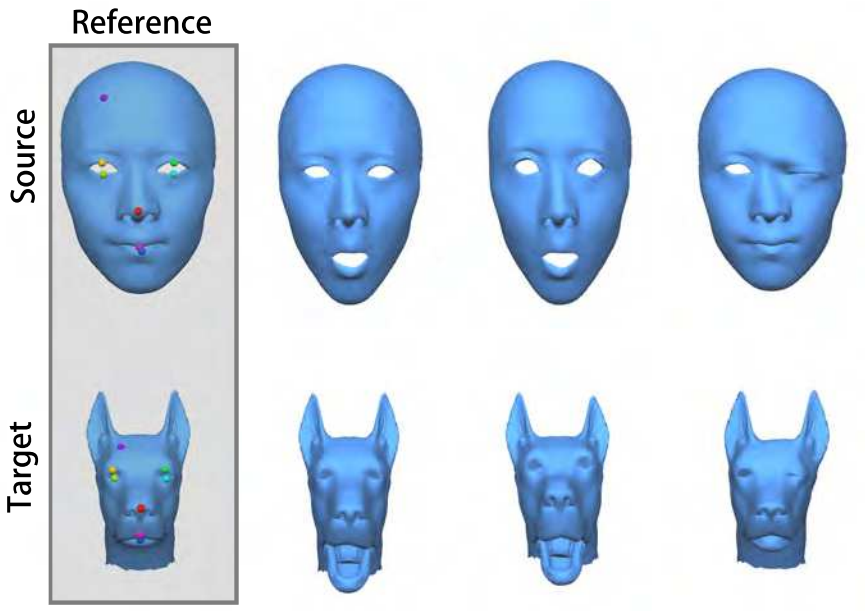


Figure 15: Deformation transfer results produced using our method showing the expressions on the face transferred to a dog. The face shapes are from the FaceWareHouse dataset.

transfer method outperforms state-of-the-art methods. We also provide an effective approach to automatically selecting key points. Extensive experiments show that this greatly reduces user effort and produces better deformation transfer results than those manually specified by normal users. Currently, our key point selection algorithm is treated as offline preprocessing. In the future we would like to consider more effective optimization approaches to speed up this stage.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No.61502453, No.61772499 and No.61611130215), Royal Society-Newton Mobility Grant (No. IE150731), the Science and Technology Service Network Initiative of Chinese Academy of Sciences (No. KFJ-STS-ZDTP-017), the Knowl-

edge Innovation Program of the Institute of Computing Technology of the Chinese Academy of Sciences under Grant No. ICT20166040.

References

[1] J. Gain, D. Bechmann, A survey of spatial deformation from a user-centered perspective, ACM Transactions on Graphics 27 (4) (2008) 107:1–107:21.

[2] D. Cohen-Or, Space deformations, surface deformations and the opportunities in-between, Journal of computer science and technology 24 (1) (2009) 2–5.

[3] R. W. Sumner, J. Popović, Deformation transfer for triangle meshes, ACM Transactions on Graphics 23 (3) (2004) 399–405.

[4] I. Baran, D. Vlasic, E. Grinspun, J. Popović, Semantic deformation transfer, ACM Transactions on Graphics 28 (3) (2009) 36:1–36:6.

[5] A. Jacobson, I. Baran, J. Popovic, O. Sorkine, Bounded biharmonic weights for real-time deformation., ACM Transactions on Graphics 30 (4) (2011) 78:1–78:8.

[6] A. Jacobson, I. Baran, L. Kavan, J. Popović, O. Sorkine, Fast automatic skinning transformations, ACM Transactions on Graphics 31 (4) (2012) 77:1–77:10.

[7] L. Gao, Y.-K. Lai, D. Liang, S.-Y. Chen, S. Xia, Efficient and flexible deformation representation for data-driven surface modeling, ACM Transactions on Graphics 35 (5) (2016) 158:1–158:17.

[8] H. Zhao, N. Lei, X. Li, P. Zeng, K. Xu, X. Gu, Robust edge-preserved surface mesh polycube deformation, Computational Visual Media 4 (1) (2018) 33–42.

[9] D. Schmitter, P. García-Amorena, M. Unser, Smooth shapes with spherical topology: Beyond traditional modeling, efficient deformation, and interaction, Computational Visual Media 3 (3) (2017) 199–215.

[10] X. Wang, S. Liu, Y. Tong, Stain formation on deforming inelastic cloth, IEEE Transactions on Visualization and Computer Graphics (2018) to appear.

[11] S.-Y. Jia, Z.-K. Pan, G.-D. Wang, W.-Z. Zhang, X.-K. Yu, Stable real-time surgical cutting simulation of deformable objects embedded with arbitrary triangular meshes, Journal of Computer Science and Technology 32 (6) (2017) 1198–1213.

[12] R. Luo, W. Xu, H. Wang, K. Zhou, Y. Yang, Physics-based quadratic deformation using elastic weighting, IEEE Transactions on Visualization and Computer Graphics (2018) to appear.

[13] X. Chen, C. Zheng, K. Zhou, Example-based subspace stress analysis for interactive shape design, IEEE Transactions on Visualization and Computer Graphics 23 (10) (2017) 2314–2327.

[14] T. Igarashi, T. Moscovich, J. F. Hughes, As-rigid-as-possible shape manipulation, ACM Transactions on Graphics 24 (3) (2005) 1134–1141.

[15] M. Colaianni, C. Siegl, J. Süßmuth, F. Bauer, G. Greiner, Anisotropic deformation for local shape control, Computational Visual Media 3 (4) (2017) 305–313.

[16] S.-Y. Chen, L. Gao, Y.-K. Lai, S. Xia, Rigidity controllable as-rigid-as-possible shape deformation, Graphical Models 91 (2017) 13–21.

[17] Y. Zhao, S. Lu, H. Qian, P. Yao, Robust mesh deformation with salient features preservation, Science China Information Sciences 59 (5) (2016) 1–9.

[18] Y.-S. Liu, H.-B. Yan, R. R. Martin, As-rigid-as-possible surface morphing, Journal of Computer Science and Technology 26 (3) (2011) 548–557.

[19] M. Alexa, D. Cohen-Or, D. Levin, As-rigid-as-possible shape interpolation, in: Proceedings of the 27th annual conference on Computer graphics and

interactive techniques, ACM Press/Addison-Wesley Publishing Co., 2000, pp. 157–164.

[20] H.-K. Chu, C.-H. Lin, Example-based deformation transfer for 3d polygon models., Journal of Information Science and Engineering 26 (2) (2010) 379–391.

[21] R. W. Sumner, M. Zwicker, C. Gotsman, J. Popović, Mesh-based inverse kinematics, ACM Transactions on Graphics 24 (3) (2005) 488–495.

[22] M. Ben-Chen, O. Weber, C. Gotsman, Spatial deformation transfer, in: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, ACM, 2009, pp. 67–74.

[23] L. Chen, J. Huang, H. Sun, H. Bao, Cage-based deformation transfer, Computers & Graphics 34 (2) (2010) 107–118.

[24] K. Zhou, W. Xu, Y. Tong, M. Desbrun, Deformation transfer to multi-component objects, Computer Graphics Forum 29 (2) (2010) 319–325.

[25] A. M. Bronstein, M. M. Bronstein, R. Kimmel, Numerical geometry of non-rigid shapes, Springer Science & Business Media, 2008.

[26] Y. Eldar, M. Lindenbaum, M. Porat, Y. Y. Zeevi, The farthest point strategy for progressive image sampling, IEEE Transactions on Image Processing 6 (9) (1997) 1305–1315.

[27] O. Sorkine, M. Alexa, As-rigid-as-possible surface modeling, in: Symposium on Geometry Processing, 2007, pp. 109–116.

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

[29] M. Botsch, M. Pauly, M. H. Gross, L. Kobbelt, PriMo: coupled prisms for intuitive surface modeling, in: Symposium on Geometry Processing, 2006, pp. 11–20.

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

[31] I. Chao, U. Pinkall, P. Sanan, P. Schröder, A simple geometric model for elastic deformations, ACM Transactions on Graphics 29 (4) (2010) 38:1–38:6.

[32] L. Liu, L. Zhang, Y. Xu, C. Gotsman, S. J. Gortler, A local/global approach to mesh parameterization, Computer Graphics Forum 27 (5) (2008) 1495–1504.

[33] M. Meyer, M. Desbrun, P. Schröder, A. H. Barr, Discrete differential-geometry operators for triangulated 2-manifolds, in: Visualization and mathematics III, Springer, 2003, pp. 35–57.

[34] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, J. Davis, SCAPE: shape completion and animation of people, ACM Transactions on Graphics 24 (3) (2005) 408–416.

[35] K. Shoemake, T. Duff, Matrix animation and polar decomposition, in: Proceedings of the conference on Graphics Interface, Vol. 92, 1992, pp. 258–264.

[36] R. M. Murray, Z. Li, S. S. Sastry, S. S. Sastry, A mathematical introduction to robotic manipulation, CRC press, 1994.

[37] G. Pons-Moll, J. Romero, N. Mahmood, M. J. Black, Dyna: A model of dynamic human shape in motion, ACM Transactions on Graphics 34 (4) (2015) 120:1–120:14.

[38] F. Bogo, J. Romero, M. Loper, M. J. Black, FAUST: Dataset and evaluation for 3D mesh registration, in: Proceedings IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2014.

[39] C. Cao, Y. Weng, S. Zhou, Y. Tong, K. Zhou, Facewarehouse: A 3d facial expression database for visual computing, IEEE Transactions on Visualization and Computer Graphics 20 (3) (2014) 413–425.