Computer-Aided Design 151 (2022) 103352

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

3D Shape Deformation Using Stick Figures

Çağlar Seylan *, Yusuf Sahillioğlu

METU, Department of Computer Engineering, Ankara, Turkey

ARTICLE INFO

Article history:
Received 16 September 2021
Received in revised form 23 April 2022
Accepted 8 June 2022

Keywords:
3D shape abstraction
Stick figure
Skeleton
Rigging
Shape deformation
ARAP

ABSTRACT

Obtaining new poses of an articulated character is a critical task in computer graphics. We address this issue with a new shape deformation approach consisting of two phases enabling the user to express the new pose as a simple stick figure, also called skeleton consisting of few bones. We interchangeably refer to this problem as shape transfer because the template shape is transferred to the independent stick figure that may be obtained from any source such as motion capture and 3D sketching. In the first phase, the stick figure is embedded into the template shape, resembling the rigging process of character skinning. Then, instead of computing blend weights as in skinning, the shape, discretized as a mesh, is augmented by adding extra edges between a subset of mesh vertices and the embedded stick figure. The second phase deforms the augmented mesh towards the new pose under the guidance of the embedded stick figure by minimizing an As-Rigid-As-Possible (ARAP) energy. The overall deformation is intuitive as skinning, preserves surface details as it is based on ARAP deformation yet has better volume preservation capability owing to the augmented mesh. Our results are validated both in terms of timing and accuracy in a comprehensive test suite that includes state-of-the-art deformation techniques.

1. Introduction

3D content creation is an important computer graphics task as many applications depend on it. To this end, one can repose an existing character or generate it from scratch based on a basic input. Our method can be seen as a mean to serve both these two purposes: We have a fixed template shape which is reposed by deformation based on an arbitrary 3D stick figure input. Since the stick figure input is independent from the template shape, one can also see our method as a from-scratch shape generator as we manage to surface this independent stick figure, a task we refer to as shape transfer. As the stick figures we use are similar to simple skeletons consisting of a few bones, stick figure and skeleton are interchangeably used throughout the article. Please note that this task corresponds to the inverse problem of skeleton extraction [1]. Countless approaches can be used to deform a mesh. Skinning (i.e., skeletal subspace deformation) methods are prominent for skeleton based deformations. In skinning, the character is rigged by a simple skeleton, called Inverse Kinematic (IK) skeleton composed of bones, and the movement of this skeleton is transferred to the surface mesh. Bone transformations are blended according to blend weights computed in rigging and then applied to each vertex of the mesh. Due to their GPU parallelism capabilities, skinning methods are widely used in real-time interactive animations like video games. Despite their fast nature, new poses in skinning are usually not accurate. Especially the skin near skeleton joints collapses towards inside of the mesh near bending regions (referred to as elbow collapse), and also an undesired artifact called candy-wrapper effect occurs near joints in bone-twist cases. Several different skinning techniques have been proposed after the introduction of LBS which is the first one of these methods [2] in order to eliminate those artifacts. Nevertheless, the accuracy of most of those techniques depends on careful adjustment of blend weights which are best done by rig artists or the selection of some shape-specific parameters. Unlike skinning, our method does not require adjustment of blend weights or any parameter.

ARAP shape deformation is another popular method for general mesh editing keeping surface details as much as possible while editing mesh surface. It is formulated for meshes embedded in 3D by Sorkine and Alexa [3]. Basically, the mesh is considered to be composed of overlapping cells. A subset of the mesh vertices are defined as control points (aka handles) and the user is allowed to move only these points. The positions of other vertices (aka free vertices) are iteratively and automatically updated such that the transformation of each cell will be composed of only rotation and translation as much as possible. ARAP deformation technique is promising for mesh reposing because the surface details are sufficiently preserved as the deformation is quite nonrigid globally, it is locally almost rigid. Another important requirement in mesh reposing is volume preservation yet original

* Corresponding author.
E-mail address: caglarseylan@gmail.com (Ç. Seylan).

https://doi.org/10.1016/j.cad.2022.103352
0010-4485/

ARAP deformation formulation does not take this into account. Several variants of ARAP deformation has been developed, which are discussed thoroughly in the supplementary document. Nevertheless, they are still not successful at volume preservation as those formulations also do not directly consider this issue.

We propose a new shape deformation approach for mesh reposing and transfer, such that, it is intuitive like skinning, that is, the deformation is driven by skeleton, and it preserves surface details as in ARAP deformations. While doing these, it produces fairly accurate results alleviating elbow collapse and candy-wrapper effects near joints, and also prevents from volume loss around articulation regions. It first embeds the target skeleton into the source mesh (referred to as the embedded skeleton) and then establishes a correspondence between the source mesh and the embedded skeleton. Finally, the source mesh is iteratively deformed towards target pose in the guidance of the embedded skeleton by using a new ARAP deformation formulation.

Potential applications of the method includes (but not limited to) the following:

- Character reposing by simple stick figures consisting of a few edges and joints. For example, it is concurrent to Gesture3D [4] for this application area. While Gesture3D reposes a character based on gesture drawings, ours does this by using 3D stick figures.
- Skin attachment to kinematic skeletons obtained by motion capture devices.
- Skin attachment to 3D skeletons which are outputs of other methods. For example, transferring a template shape to the skeletons obtained by 3D pose estimation methods from 2D images.

Results and discussions for usage of our method on the mentioned applications above are given in Section 6. Our contributions in this study are as follows:

- A user-friendly framework to create new poses of an articulated human or non-human character with two basic inputs: a stick figure as target (from any source such as motion capture, sketching, [5]) and the source template shape (to be referred as the source mesh from now on).
- A skeleton embedding approach which can also be used as an automatic rigging method.
- A method to augment the source mesh for a novel volume-preserving ARAP deformation.

A brief literature review about automatic rigging, ARAP deformation and deformation transfer is given in Section 2. In Section 3, we provide an explanation of the overall framework. In Sections 4 and 5, we explain the skeleton embedding and our deformation framework, respectively. In Section 6, we provide both qualitative and quantitative results of our approach and the discussion about them. The conclusion and future work is provided in Section 7.

2. Related work

2.1. Skeleton embedding

Target skeleton is embedded into the source mesh in the first phase of our approach. Most of the previous work about skeleton embedding is related to automatic rigging methods which include both embedding IK skeleton into source mesh and determining blend weight of each mesh vertex.

We have to emphasize that computing an IK skeleton from scratch and embedding an IK skeleton inside of the mesh are different and only the latter one serves our purpose. A well-known example to the former one was proposed by Aujay et al. [6] which constructs the Reeb graph of a mesh by using a Reeb graph computation method [7], and then embeds this graph into the 3D mesh. Machine learning can also be used other than analytical methods for this purpose, a recent approach uses deep neural networks to predict animation skeletons for 3D articulated models [8]. Even though these methods produce appealing IK skeletons, they are not suitable for our purpose. We need the embedded version of the target skeleton defined by the user, i.e., the embedded skeleton should have the same number of joints and edges, and the connection of these primitives should exactly be same with the target skeleton.

A related approach proposed by Baran and Popović [9] constructs a connected graph using medial surface of the input mesh such that each node represents center of a maximal ball. The input skeleton is embedded onto the graph by minimizing a penalty function composed of basis functions. Learning procedure is required to identify the optimum coefficients of the basis functions. The identification process was intervened manually. As a requirement, the input skeleton and mesh should be in a similar orientation, otherwise the embedding result is wrong. On the contrary, our embedding method is completely orientation agnostic, and no coefficient learning procedure is required.

Instead of trying to fit the input skeleton onto a complex graph, matching it with the curve-skeleton of the input mesh is more promising, which is also what we do in this study. Pourier et al. [10] adopted a similar strategy. They first extract the curve-skeleton using Reeb graph which combines the methods in [6,11]. Then the input skeleton is matched with the curve skeleton using a sub-tree based shape descriptor. The approach can potentially produce good embeddings yet some parameters called length variation threshold, and weight balancing parameter should be tuned by the user, and apparently they seem to be different for each different mesh.

Another IK skeleton embedding approach based on matching with curve-skeleton is proposed by Pantuwong et al. [12]. As their matching method is based on distance field, their extraction algorithm is based on voxel mesh [13] forcing the user to make a choice between quality and timing. Also the predefined IK skeleton should be attached with semantic information which increases user labor.

A recent IK skeleton embedding approach [14] also matches the input skeleton with the curve skeleton of the input mesh extracted by mesh contraction [15]. Although the method does not require user involvement, it is only specific for human body models which seriously limits its usage. Stick figures in the form of skeletons are used to repose non-human shapes like chairs [16]. Deep learning techniques also enable reposing of 3D human bodies via 2D stick figures [17].

2.2. Skinning

The second stage of our approach resembles skinning methods in the sense that skin deformation of a character is computed from the movement of bones to which the skin is attached. LBS [2] deforms vertex positions by linearly blending bone transformations according to the bone influence weights. However, linearly combining rigid transformations violates orthogonality principle [18] which causes volume-loss artifacts near joints known as elbow-collapse in bending and candy-wrapper effect in twisting. To remedy this issue, spherical blending of transformations [19] and usage of dual-quaternions [20] were proposed yet these approaches introduced bulging artifacts near joints.

Several interesting skinning methods have been developed to alleviate those artifacts over time. For example, Rohmer et al. [21] proposed a method that exactly preserves volume of the shape. However, this method requires additional inputs like amount of

corrections and profile curves to control the volume preservation which does not fit our requirements. Realistic results can be obtained with physically-based skinning methods like [22] but those methods also need some parameters like Young's modulus and Lamé parameters which differ from shape to shape. It is also possible to learn optimum blend weights by using results of accurate physically-based methods [23]. By introducing new deformers at joints in addition to bones, accurate results can be obtained but still the parameters of physically-based methods should be tuned carefully for the blend weight learning stage.

Jacobson and his colleagues [24] proposed a fast skinning method that integrates hierarchical bone structure and ARAP framework. The method reduces to standard LBS when adapted to our framework without abstract handles. Using abstract handles according to the provided algorithm improves the result but still some previously mentioned artifacts remain (comparisons with this method are provided in Section 6).

Le et al. [25] proposed to pre-compute optimum rotation centers of the mesh vertices before deformation. However, this method requires careful setting of blend weights which also contradicts with our design requirements. Moreover, the method also causes some volume loss artifacts around places where multiple bones meet like chest.

Other than direct skinning methods, implicit surface based methods may produce high quality results but the quality of the deformation depends on the accuracy of the initial skinning solution [26]. Also, the final pose depends both on the skeleton pose and on the whole animation history, which causes the final pose to be affected by chosen time step [27]. Moreover, those methods require setting of many parameters, adjustment of which may require a few experiments.

In fact, all the methods using blend weights and/or requiring setting of several parameters are not suitable for our framework because performance of those methods highly depends on the blend weights and those parameters. For example, manual adjustment of blend weights by rig artists gives better results than computing them with automatic methods like [9]. Nonetheless, we do not provide any other input to our framework other than the source mesh and target skeleton. There are some recent skinning methods like [28] not requiring any blend weight to compute the deformed shape. Although appealing results can be obtained with this method, some artifacts still remain especially in highly deformed cases (comparisons with this method are provided in Section 6).

With the advents and great progress in deep learning methods in recent years, neural network based shape deformation [29,30] and skinning methods [31,32] are also becoming popular. Li and his colleagues [32] introduced Neural Blend Shapes (NBS) recently, which solves a problem similar to ours, i.e., deforming a shape in the guidance of a simple skeleton. Although it is possible to obtain high quality and natural deformations with such methods, they adopt the drawback of supervised learning methods: Their performance is highly dependent on the train set, which should include a rich set of example mesh-skeleton pairs covering the set of common deformations in articulated characters. This fact makes the training especially difficult for non-human characters. On the other hand, owing to being based on well-known ARAP framework, our method does not require any example mesh-skeleton pair or training process but still able to obtain deformations with comparable quality. Comparisons of our method with NBS and more discussions are provided in Section 6.

2.3. Deformation transfer

When we have more than one target skeleton, our problem can be seen as a special case of deformation transfer: We need to obtain new poses of a shape by using the deformation of skeletons. Key point selection and establishing correspondence is one of critical tasks in deformation transfer. Sumner et al. gave this task to the user in their pioneering work [33]. In [34], Yang et al. proposed an automatic algorithm to select key points automatically but defining corresponding points was still the responsibility of the user. Baran et al. showed it is also possible to capture the deformation semantically by extracting the correlation between several source-target shape pairs [35]. Gao et al. proposed a Generative Adversarial Network (GAN) based deformation transfer method which requires no user intervention for key point selection and correspondence establishment [29]. We do not transfer a deformation from mesh to mesh in our problem, instead we guide deformation by using skeletons. For this reason, there is no concept of key point selection and correspondence computation in our problem as in deformation transfer. To read more on deformation transfer, we redirect our readers to the comprehensive survey [36].

3. Method overview

The method takes two inputs: a closed 2-manifold triangular source mesh, and a target skeleton which is a 1D connected graph embedded in 3D representing the pose to which the source mesh will be deformed. Its output is a closed 2-manifold triangular mesh representing the new pose of the source mesh dictated by the target skeleton such that the target skeleton is embedded inside of the output mesh. Overall approach consists of two consecutive phases and summarized in Fig. 1.

The target skeleton is embedded into the source mesh in the first phase. To do this, we first extract curve-skeleton of the source mesh by using Mean Curvature Flow (MCF) approach [15]. Our genetic algorithm based graph matching method matches the target skeleton with the curve-skeleton, embedding the target skeleton onto the curve-skeleton in essence. The output of the first phase is the source mesh with this embedded skeleton in it. Details of the skeleton embedding are given in Section 4.

At the beginning of the second stage before deformation, the source mesh is augmented. More specifically, the embedded skeleton is sampled and additional edges are added between these sample points and a subset of mesh vertices, which we call support edges. Then the shape is deformed both by using the point-by-point registration between the embedded skeleton and the target skeleton and by minimizing the ARAP energy defined for the augmented mesh. While ARAP deformation scheme preserves the surface details during deformation, the support edges prevent volume loss and undesired artifacts near bending joints. Details of the second phase are briefly explained in Section 5.

4. Target skeleton embedding

4.1. Dissimilarity of two skeletons

We define a measure indicating dissimilarity between two curve-skeletons to match the target skeleton with the extracted skeleton. Let $d_{\mathcal{S}}(\mathbf{x})$ be the distance between a point $\mathbf{x}$ and the closest point on curve-skeleton $\mathcal{S}$ to $\mathbf{x}$. $d_{\mathcal{S}}(\mathbf{x})$ defines a scalar field such that $d_{\mathcal{S}} : \mathbb{R}^3 \rightarrow \mathbb{R}$. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two curve-skeletons. As $d_{\mathcal{S}}(\mathbf{x})$ is a scalar field, curve integral of $d_{\mathcal{S}_2}(\mathbf{x})$ over $\mathcal{S}_1$ gives us a measure of how dissimilar is $\mathcal{S}_1$ from $\mathcal{S}_2$. If we denote

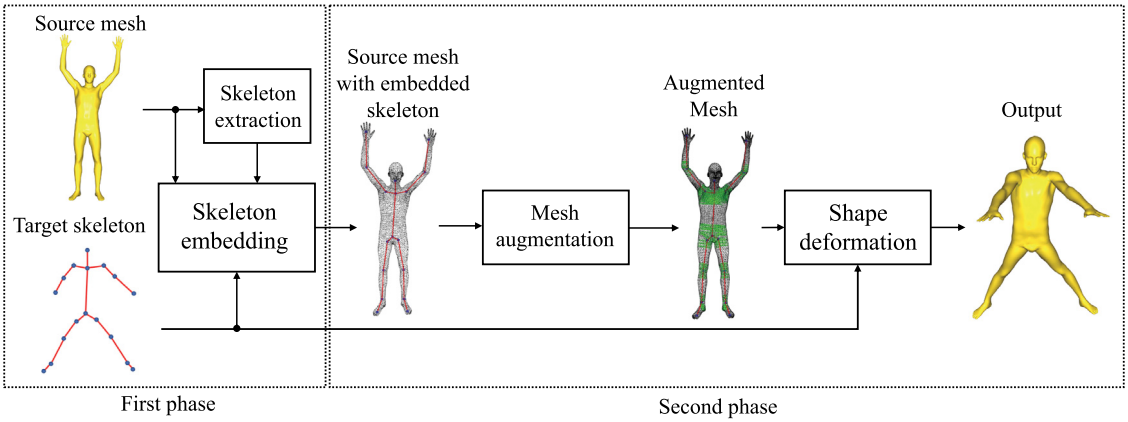


Fig. 1. The overall method consists of two stages. The first phase takes a source mesh and a target skeleton, and outputs the source mesh with the embedded version of the target skeleton inside of it. The second phase takes the source mesh with embedded skeleton and the target skeleton as inputs. It augments the source mesh by adding additional edges and deforms the shape towards the target skeleton by using ARAP deformation.

dissimilarity of $\mathcal{S}_1$ from $\mathcal{S}_2$ as $D_{\mathcal{S}_2}(\mathcal{S}_1)$, it can be computed with the formula,

$$D_{\mathcal{S}_2}(\mathcal{S}_1) := \int_{\mathcal{S}_1} d_{\mathcal{S}_2}(\mathbf{r})ds \tag{1}$$

where $\mathbf{r}$ is a bijective parametrization of $\mathcal{S}_1$ and ds is infinitesimally small curve element. If we uniformly choose N sample points on curve-skeleton $\mathcal{S}_1$, then Eq. (1) can also be written as

$$D_{\mathcal{S}_2}(\mathcal{S}_1) = \lim_{N\to\infty}[\frac{1}{N}\sum_{i=0}^{N} d_{\mathcal{S}_2}(\mathbf{s}_i)] \tag{2}$$

where $\mathbf{s}_i$ is the ith sample point. Our approximation to Eq. (1) is based on Eq. (2).

Notice that $D_{\mathcal{S}_2}(\mathcal{S}_1)$ is non-commutative. This can lead to non-intuitive results. For example in case $\mathcal{S}_2 \subset \mathcal{S}_1$, $D_{\mathcal{S}_2}(\mathcal{S}_1) = 0$ and $D_{\mathcal{S}_1}(\mathcal{S}_2) > 0$ yet it is clear that $\mathcal{S}_1$ and $\mathcal{S}_2$ are not similar and the latter one is correct. We define commutative dissimilarity function D as

$$D(\mathcal{S}_1, \mathcal{S}_2) := D_{\mathcal{S}_2}(\mathcal{S}_1) + D_{\mathcal{S}_1}(\mathcal{S}_2), \tag{3}$$

on which our skeleton matching algorithm is based.

4.2. Skeleton matching

A skeleton is represented in our discrete setting as $\mathcal{S} = (V, E)$ where V is the set of vertices and E is the set of edges connecting these vertices. $\mathcal{S}_{trg} = (V_{trg}, E_{trg})$ denotes the target skeleton, $\mathcal{S}_{cs} = (V_{cs}, E_{cs})$ denotes the extracted curve-skeleton of the source mesh, and $\mathcal{S}_{emb} = (V_{emb}, E_{emb})$ denotes the embedded version of the target skeleton. There are three types of vertices: terminal, joint, and regular. A vertex with only one neighbor is called a terminal vertex, with exactly two neighbors is called a regular vertex, and with more than two neighbors is called a joint vertex. A sequence of line segments connected with only regular vertices between joint or terminal vertices is called a skeleton segment.

The definitions above are valid for both $\mathcal{S}_{cs}$ and $\mathcal{S}_{trg}$. To embed $\mathcal{S}_{trg}$ into the source mesh we match terminal vertices of $\mathcal{S}_{trg}$ with terminal vertices of $\mathcal{S}_{cs}$ and joint vertices of $\mathcal{S}_{trg}$ with joint vertices of $\mathcal{S}_{cs}$. We place a regular vertex of $\mathcal{S}_{trg}$ on $\mathcal{S}_{cs}$ by keeping the ratio of geodesic distances from it to the endpoints of the segment it lies on. More specifically, let us consider the setup in Fig. 2, suppose $\mathbf{v}_j$ matches with $\mathbf{v}'_j$ and $\mathbf{v}_t$ matches with $\mathbf{v}'_t$. We know $\mathbf{v}_r$ and want to compute $\mathbf{v}'_r$. Let $g(\mathbf{v}_1, \mathbf{v}_2)$ be the geodesic distance between $\mathbf{v}_1$ and $\mathbf{v}_2$, let $\mathbf{r}'_{seg}(t)$ be the parametric representation of the skeleton segment belonging to the embedded skeleton such that $\mathbf{r}'_{seg}(0) = \mathbf{v}'_j$, $\mathbf{r}'_{seg}(1) = \mathbf{v}'_t$. Then, $\mathbf{v}'_r$ is computed as

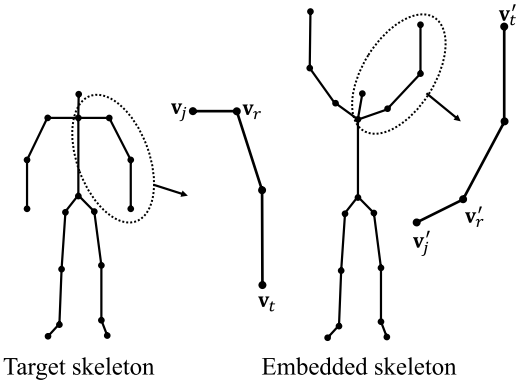


Fig. 2. A skeleton segment in a target skeleton and its embedded version. $\mathbf{v}_j$ and $\mathbf{v}_t$ are the end points of the segment and $\mathbf{v}'_j$ and $\mathbf{v}'_t$ are their corresponding points in the embedded version. The position of $\mathbf{v}'_r$ is determined according to the parametric position of $\mathbf{v}_r$.

$$\mathbf{v}'_r = \mathbf{r}'_{seg}(\frac{g(\mathbf{v}_j, \mathbf{v}_r)}{g(\mathbf{v}_j, \mathbf{v}_t)}). \tag{4}$$

The mentioned matching rules are necessary but not sufficient to obtain a valid embedding. Fig. 3 depicts one valid and one invalid embedding both satisfying the rules above.

The embedded skeleton should be similar to the extracted skeleton as much as possible, that is, no other matching should give smaller value than $D(\mathcal{S}_{emb}, \mathcal{S}_{cs})$ according to Eq. (3). We state the optimization problem as follows:

$$\mathcal{S}_{emb} = \arg\min_{\mathcal{S}} D(\mathcal{S}, \mathcal{S}_{cs}) \tag{5}$$

such that,

- $E = E_{trg}$,
- Joint vertices of $\mathcal{S}$ match with only joint vertices of $\mathcal{S}_{cs}$,
- Terminal vertices of $\mathcal{S}$ match with only terminal vertices of $\mathcal{S}_{cs}$,
- Regular vertices of $\mathcal{S}$ are placed according to Eq. (4).

This is a constrained non-linear discrete optimization problem with no concept of gradient. Moreover, some parts of an embedding may belong to the optimal solution. These observations as well as the recent success of the evolutionary methods on shape

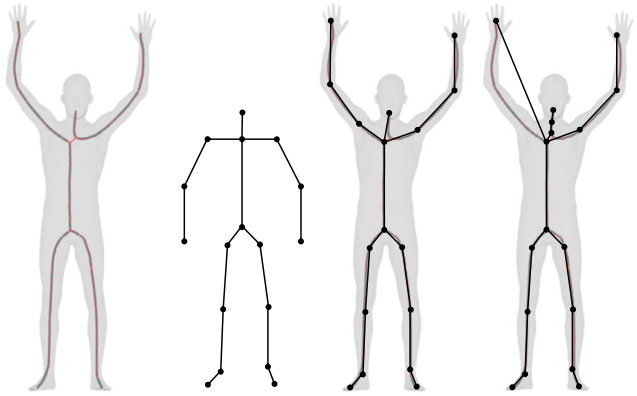


Fig. 3. First column shows the extracted curve-skeleton of a human mesh to be used as the source. Second column shows a target skeleton. Third column shows a correct matching between the extracted skeleton and the target skeleton. Fourth column is an example to an invalid matching.

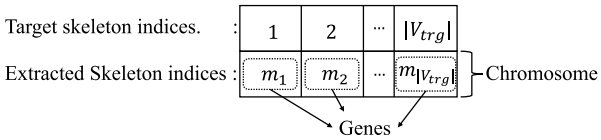


Fig. 4. Chromosomes and genes in the skeleton matching problem.

matching [37] suggest us to use genetic algorithm optimization framework for skeleton matching.

Let indices of target skeleton vertices be $(1, 2, \ldots, |V_{trg}|)$ and indices of the extracted curve-skeleton vertices to which they are matched be $(m_1, m_2, \ldots, m_{|V_{trg}|})$. In this representation, ith vertex of the target skeleton matches with m_ith vertex of the extracted skeleton. Regular vertices are not matched and represented by -1, they instead placed according to Eq. (4) after joint and terminal vertices are matched. Chromosome and gene definitions are shown in Fig. 4.

We choose the fitness function f of chromosome χ to be

$$f(\chi) = \frac{1}{D(chrom2Skel(\chi), \mathcal{S}_{cs})} \tag{6}$$

where $chrom2Skel(\chi)$ is a procedure converting chromosome χ to the skeleton represented by χ.

We chose initial population size to be 40 where chromosomes are initialized randomly obeying terminal-terminal and joint-joint matching rule. The Partially-Mapped Crossover (PMX) [38] strategy is used for crossover operation. During crossover, terminal vertices are matched only with terminal vertices and joint vertices are matched with only joint vertices. Two parents are selected with fitness proportionate selection strategy and two offsprings are produced in each crossover. The population is same during the optimization process, i.e., if a crossover happens, two victims are selected by using inverse fitness proportionate selection. Only one randomly selected gene is swapped in the mutation operation which may follow crossover operation. A terminal vertex is swapped with a terminal vertex, and a joint vertex is swapped with a joint vertex in the mutation. Probabilities of crossover and mutation operations to be applied in a single iteration are 0.8 and 0.4, respectively.

A sample run of skeleton matching is summarized in Fig. 5. Here, a target skeleton for a wolf mesh consisting of 19 vertices is matched with the curve-skeleton of the source mesh consisting of 508 vertices. Fitness function value of the fittest chromosome for each iteration is also provided as a graph in the same figure. The algorithm converged to the optimum solution in around 200 iterations.

Both per iteration and total running time of the skeleton embedding algorithm (using double-precision floating-point variables) for 500 iterations are given in Table 1. The running time can well be improved by using a parallel algorithm in multi core setting, and by using single-precision floating-point variables. It should also be noted that run time of genetic algorithms are highly dependent on the parameters such as population size.

4.3. Limitations and discussion

One limitation of our skeleton matching method is that it is not guaranteed that the genetic optimization will result in the embedding exactly the user desires. For example, imagine a source mesh with two legs and a tail whose lengths are all equal and the configurations of the regular vertices are same. The optimization may end up an embedding in which one of the legs and the tail may be swapped. We allow the user to correct such embeddings by manually swapping some of the skeleton segments in a drag-and-drop manner after the optimization process ends. Also, vertex positions of a rigged skeleton often depend on the user's or artist's decision and it is rather subjective [8]. Thus, we also allow the user to change the positions of the vertices of the embedded skeleton after the optimization ends. However, please not that we did not alter the vertex positions of the embedded skeletons manually in all of the experiments in this study.

One question arising in this context is how would the skeleton matching method behave if there is a topological difference between the target and extracted skeleton. If the number of terminal vertices of the target and extracted skeletons are different, the genetic algorithm still does the matching by minimizing Eq. (3) and leaves some terminal vertices unmatched. An example to this case is depicted in Fig. 6. However, if the number of loops in the target and extracted skeleton are different, the algorithm fails to match the skeletons. So we require the genera of the target and extracted skeletons be the same.

5. Mesh transfer

The second phase of our approach is mesh transfer. Like the first one, this phase can also be used on its own. Before deforming the shape to obtain new poses, the mesh should be augmented by adding extra edges between its free vertices and the skeleton. Herein, one may be curious about why we add extra edges instead of defining an ARAP energy like,

$$\begin{aligned} E(\mathcal{M}, \mathcal{M}') = \sum_{i=1}^{|V|} \sum_{j \in \mathcal{N}(i)} & w_{ij} \|(\mathbf{p}_i' - \mathbf{p}_j') - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)\|^2 \\ & + w_{is} \|d_{\mathcal{S}_{emb}}(\mathbf{p}_i) - d_{\mathcal{S}_{trg}}(\mathbf{p}_i')\|^2 \end{aligned} \tag{7}$$

where $d_{\mathcal{S}_{emb}}(\mathbf{p}_i)$ is the closest distance of $\mathbf{p}_i$ to the embedded skeleton, $d_{\mathcal{S}_{trg}}(\mathbf{p}_i')$ is the closest distance of $\mathbf{p}_i'$ to the target skeleton and w_{is} is an appropriate weight. Nevertheless, such an energy equation cannot be solved for $\mathbf{p}_i'$'s as in the original ARAP. However, inserting such pointwise correspondences between the mesh and its skeleton as in-between edges into the energy equation results in a much simpler optimization problem.

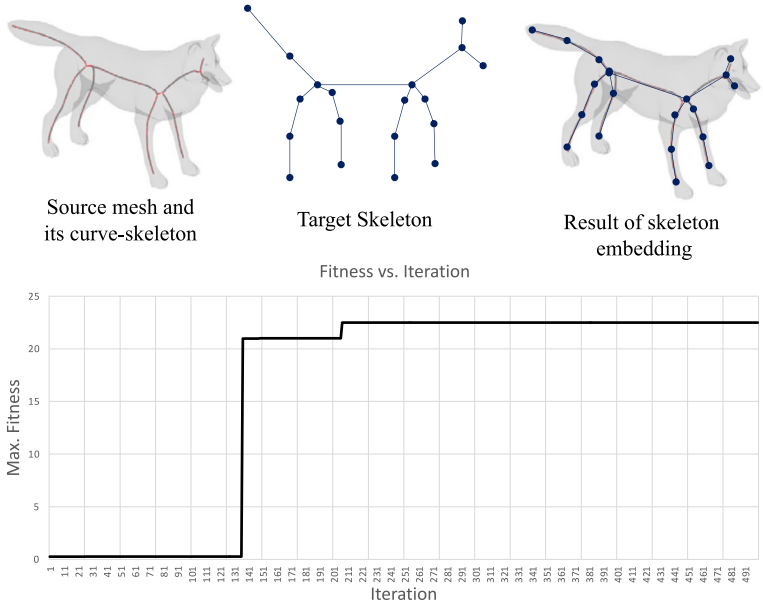


Fig. 5. A sample run of skeleton matching method based on genetic algorithm framework. While the target skeleton consists of 19 vertices, the curve-skeleton consists of 508 vertices. The algorithm converges to the optimum solution in around 200 iterations, as can be seen from the fitness vs. iteration graph.

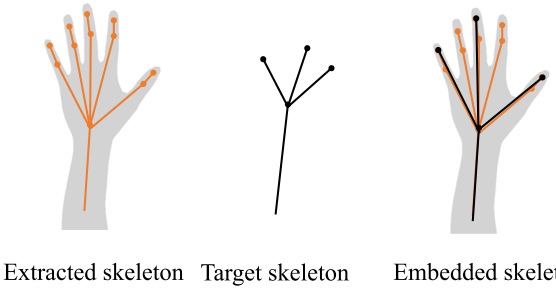


Fig. 6. A case in which the number of terminal vertices between the extracted and target skeletons are different. The genetic algorithm matches right finger of the target skeleton with the rightmost finger of the extracted skeleton in this case. In a similar fashion, left finger is matched with the leftmost finger, and middle finger is matched with again the middle finger.

5.1. Selection and movement of handle vertices

In ARAP deformation, a subset of the mesh vertices are selected as handle vertices. The user is only allowed to move those handle vertices freely. Positions of non-handle, also called free vertices, are updated automatically by minimizing the ARAP energy. It is suitable to choose the parts of the mesh faraway to joints as handles as they are almost rigid during the deformation. The movement of those handle vertices are dictated by the skeleton. Thus, we first establish a correspondence between the skeleton and mesh to decide which bones will move which handles.

Each mesh vertex is corresponded with the closest point to it on the skeleton. We assign a parametrization $\mathbf{r}_{bone}(t), t \in [0, 1]$ to each bone, such that, $\mathbf{r}_{bone}(0) = \mathbf{u}$ and $\mathbf{r}_{bone}(1) = \mathbf{v}$ where $\mathbf{u}$ and $\mathbf{v}$ are coordinates of two end points of the bone which makes each corresponding point of a mesh vertex has a parametric coordinate. If the length of a bone with joint vertices in both ends is less than the average bone length in the skeleton, we choose all mesh vertices corresponding to that bone as free vertices. For other bones, handle selection procedure is controlled with a parameter $\rho \in [0, 1]$. Let t_i be the parametric coordinate of the corresponding point of mesh vertex $\mathbf{p}_i$ on a bone. We have three possibilities for the bone:

1. If neither $\mathbf{u}$ nor $\mathbf{v}$ is terminal, and if $t_i \in [0.5 - \rho/2, 0.5 + \rho/2]$, then we define $\mathbf{p}_i$ as handle.
2. If $\mathbf{u}$ is terminal but $\mathbf{v}$ is not terminal, and if $t_i \in [0, 0.5 + \rho/2]$, then we define $\mathbf{p}_i$ as handle.
3. If $\mathbf{u}$ is not terminal but $\mathbf{v}$ is terminal, and if $t_i \in [0.5 - \rho/2, 1]$, then we define $\mathbf{p}_i$ as handle.

Those three cases are depicted in 2D in Fig. 7 where black points represent handles and green points represent free vertices. With this way the movements of only the mesh vertices near joints will be updated by the minimization of ARAP energy.

Bones in the embedded skeleton are moved to their positions in the target skeleton one by one. The handles corresponding to the points on a bone are moved to the target in the same way as that bone.

5.2. Mesh augmentation

We call the additional edges between some sample points on the embedded skeleton and free vertices of the source mesh as support edges. We call the mesh updated with the support edges as the augmented mesh.

At the beginning of mesh augmentation we uniformly choose sample points on the embedded skeleton where the gap between each is ϵ. If we denote ith sample as $\mathbf{s}_i$, we compute normal plane $\mathcal{P}_i$ of the embedded skeleton at $\mathbf{s}_i$ as shown at left of Fig. 8. Then we mark the mesh vertices whose distances to $\mathcal{P}_i$ are smaller than $\epsilon/2$ and which are free as candidates. With this way we actually have chosen as candidate all the free vertices of the mesh between two parallel planes which are also parallel to $\mathcal{P}_i$, and distance between $\mathcal{P}_i$ and both are $\epsilon/2$. Those two planes can be seen at right of Fig. 8. We chose ϵ to be the average edge length of the source mesh throughout the study.

For each candidate, we check whether the edge between it and its corresponding vertex on the skeleton intersects the mesh. If it does, we discard the candidate because support edges should completely be inside of the source mesh. If the sample points on the skeleton are incident to the edges supporting unrelated parts

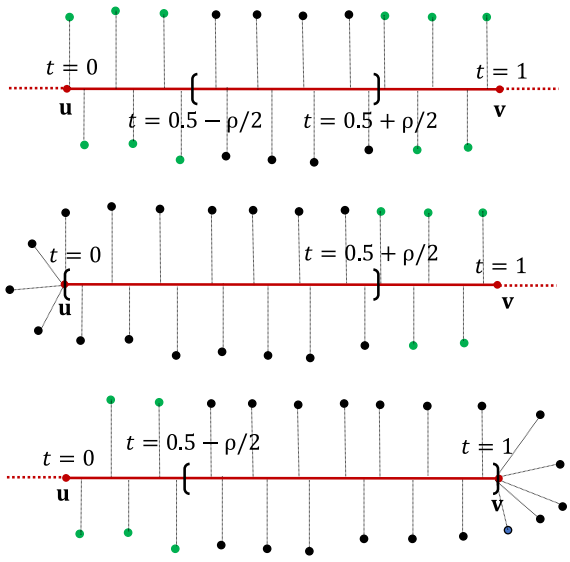


Fig. 7. Three cases for a bone while selecting free vertices according to ρ in a 2D case. In the top, neither **u** nor **v** is terminal, in the middle **u** is terminal but **v** is not terminal, in the bottom, **u** is not terminal but **v** is terminal. Green points represent free vertices and black points represent handles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

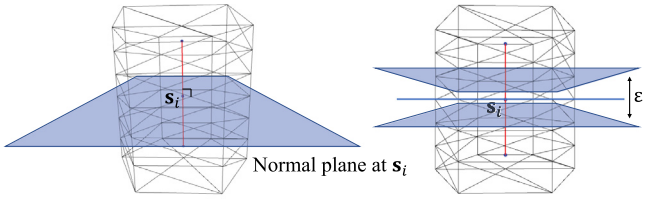


Fig. 8. Left figure shows the normal plane at point $\mathbf{s}_i$ of the skeleton, which is shown as a red stick, inside of a hexagonal prism. Tight figure shows two parallel planes to the normal plane $\mathbf{s}_i$ with ϵ gap in-between.

of the mesh, the transfer can result in inconsistent deformations. Thus, all the support edges emanating from a sample should support the vertices only in the most related part to it. To find the most related mesh part of a sample point, we paint each disconnected free vertex set (They are disconnected by handles.) to a different color. The most related part of a sample point is the free vertex set having the color of the closest one of remaining candidates to it. We then discard the candidates in other than the most related parts.

Some mesh vertices may still be corresponded to more than one skeleton sample after selecting candidates for each sample point in the mentioned way. Multiple support edges may cause inconsistent deformations around neighborhood of these vertices. Thus, we choose the closest corresponded skeleton sample in multiple correspondence cases and discard the others. We add edges between all remaining corresponding pairs of free mesh vertices and skeleton samples. The support edges for a human mesh are shown in Fig. 9. Free vertices of a mesh are indicated with green color and handles are indicated with yellow color both in Fig. 9 and all other figures in the paper.

5.3. Skeleton-driven ARAP deformation

Let us suppose the source mesh is represented as $\mathcal{M}_{src} = (V_{src}, E_{src})$ where V_{src} is the vertex set and E_{src} is the edge set. Let the set of support edges be E_{sup} and additional vertices incident

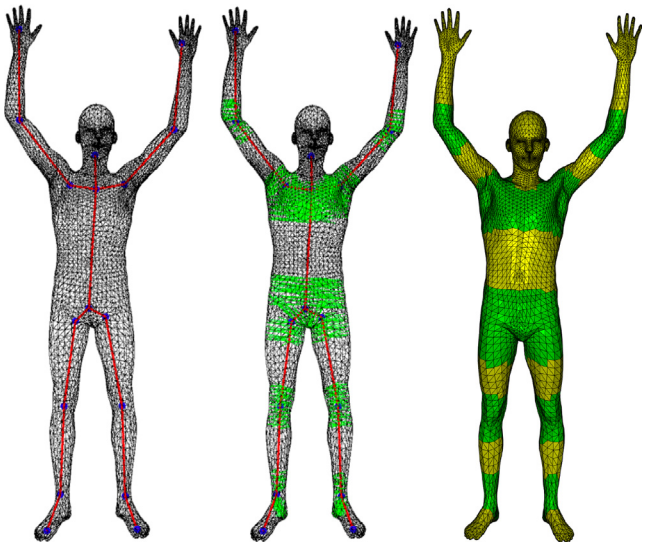


Fig. 9. Left subfigure shows the skeleton embedded into the human mesh. Middle subfigure shows support edges added to the source mesh which is the augmented mesh. Right subfigure shows the source mesh, green regions represent free vertices, yellow regions represent handles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

to support edges be V_{sup}. Then the augmented mesh can be represented as $\mathcal{M}_{aug} = (V_{aug}, E_{aug})$ where $V_{aug} = V_{src} \cup V_{sup}$ and $E_{aug} = E_{src} \cup E_{sup}$.

The local cell $\mathcal{C}_i$ corresponding to vertex i consists of the vertex i itself and its 1-ring neighborhood $\mathcal{N}(i)$. The local cell is defined over the augmented mesh, which means a skeleton edge may be incident to a mesh vertex i. Also, there are no edges between vertices in V_{sup}.

If we represent local rigidity energy per cell as in [3]

$$E(\mathcal{C}_i, \mathcal{C}_i') = \sum_{j \in \mathcal{N}(i)} w_{ij} \|(\mathbf{p}_i' - \mathbf{p}_j') - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)\|^2 \tag{8}$$

where $\mathcal{C}_i$ is the cell of ith vertex and $\mathcal{C}_i'$ is its deformed version. Choosing weights of support edges are tricky here. As $\mathcal{M}_{aug}$ is not manifold anymore, we cannot compute them with the well known cotangent formula. The weight of the support edge incident to ith vertex should be related to the weights of non-support edges incident to it. Choosing the weight of the support edge as the average of weights of non-support edges produce satisfying and consistent results, i.e., deformation process both preserves surface details and prevents the surface from collapsing inwards. Thus, edge weights are computed with the following formula:

$$w_{ij} = \begin{cases} \frac{1}{2}[cot\alpha_{ij} + cot\beta_{ij}] & \text{if } (i,j) \in E_{src} \\ \frac{1}{2|\mathcal{N}'(i)|} \sum_{k \in \mathcal{N}'(i)} [cot\alpha_{ik} + cot\beta_{ik}] & \text{if } (i,j) \in E_{sup} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where α_{ij} and β_{ij} are the opposite angles of edge (i,j) and $\mathcal{N}'(i) = \mathcal{N} - \{j\}$.

The ARAP energy is defined as in [3] and computed by using the augmented mesh and its deformed version $\mathcal{M}'$. Namely,

$$E(\mathcal{M}_{aug}, \mathcal{M}_{aug}') = \sum_{i=1}^{N_{aug}} \sum_{j \in \mathcal{N}(i)} w_{ij} \|(\mathbf{p}_i' - \mathbf{p}_j') - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)\|^2 \tag{10}$$

where N_{aug} is the number of vertices in the augmented mesh. Notice that we implicitly set the weight of per cell rigidity energy to 1 as suggested.

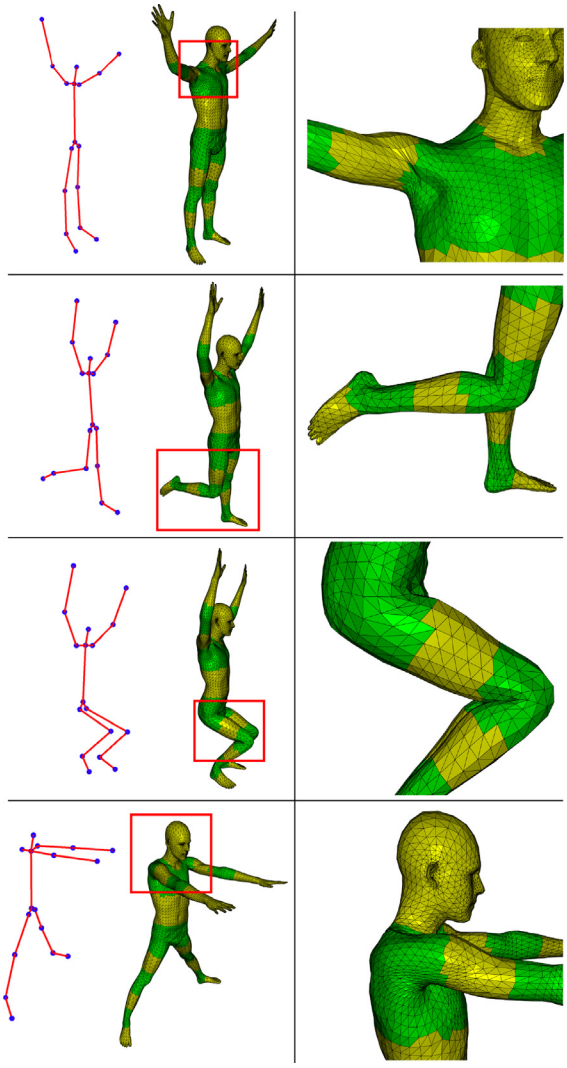


Fig. 10. Various poses generated by deforming the source mesh in Fig. 9 towards the target skeletons shown in the left column with the resulting meshes. Right column zooms to the highlighted regions on the results.

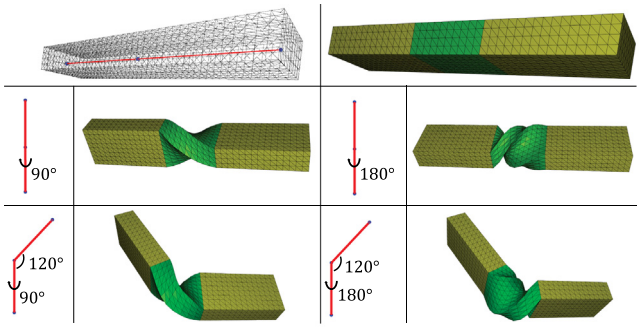


Fig. 11. Twisting a box by rotating the skeleton edge as shown. The embedded skeleton and source mesh are given in the top row. Middle row shows the results of rotating the edge for 90° and 180°, The last row shows the results of defining 60° bending in addition to the rotations.

Solution of Eq. (10) is approximated iteratively where each iteration consists of two steps. In the first step, $\mathbf{p}_i'$'s are used from the previous step and optimum $\mathbf{R}_i$'s minimizing $E(\mathcal{C}_i, \mathcal{C}_i')$ in Eq. (8) are computed. Computation of optimum $\mathbf{R}_i$'s is briefly explained in [39] and we will not repeat the same explanation here. In the implementation, we used McAdams' fast SVD algorithm for 3 by 3 matrices to compute optimum $\mathbf{R}_i$'s [22].

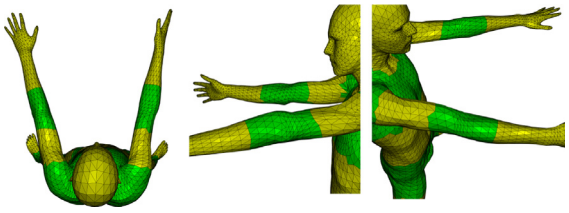


Fig. 12. Rotating the right arm of a human around its corresponding bone without suffering from the candy-wrapper artifact. Result shown in different views.

After computing optimum local rotations in the first step, optimum $\mathbf{p}_i'$'s minimizing $E(\mathcal{M}_{aug}, \mathcal{M}'_{aug})$ in Eq. (10) are computed. Taking derivative of Eq. (10) w.r.t. $\mathbf{p}_i'$ and doing the required arrangements results in the linear system

$$\mathbf{L}_{aug}\mathbf{p}' = \mathbf{b}. \tag{11}$$

$\mathbf{L}_{aug}$ in Eq. (11) differs from the system matrix in original ARAP deformation in the sense that $\mathbf{L}_{aug}$ contains weights for support edges. Let $\mathcal{H}$ be the set of indices of handle vertices, i.e., the vertices computed with the procedure explained in Section 5.1 and skeleton samples incident to a support edge. The elements l_{ij} of $\mathbf{L}_{aug}$ are computed as,

$$l_{ij} = \begin{cases} w_{ij} & \text{if } i \neq j \text{ and } i \notin \mathcal{H} \\ -\sum_{j \in \mathcal{N}(i)} w_{ij} & \text{if } i = j \text{ and } i \notin \mathcal{H} \\ 1 & \text{if } i = j \text{ and } i \in \mathcal{H}. \end{cases} \tag{12}$$

If $i \in \mathcal{H}$ then $\mathbf{b}_i$ is set to the position of ith vertex which is known.

We first pre-factored the system matrix of the sparse linear system in Eq. (11) by LU decomposition then solved the system for $\mathbf{p}'_x$, $\mathbf{p}'_y$, and $\mathbf{p}'_z$ independently by using Eigen linear algebra library in C++. The iteration is stopped when $(E' - E)/E < 10^{-5}$ where E' and E are the global rigidity errors in the current iteration and previous iteration, respectively. Please note that, we moved each vertex of the source mesh by applying the transformation of the closest bone (rigid skinning) prior to the iterative solution, and used this shape as the initial solution.

6. Results and discussion

Visual results. Different poses of the human figure from MPI FAUST dataset [40] in Fig. 9 generated by our method can be seen in Fig. 10. Target skeletons and new poses can be seen in the left column and zoomed regions highlighted in the right column. It can be verified from the figure that surface details are preserved around armpits, shoulders, ankles, waist and knee during the deformation.

The user can define rotation around the axis of a target skeleton edge. The effect is formed as a twist near joint regions on the surface. An example is given in Fig. 11. The embedded skeleton and source mesh of a box is given in the top row. Resulting deformations for 90° and 180° rotations are given in the middle row. Deformation results of adding 60° to the target skeletons are given in the last row. A result of applying rotation around skeleton edge axis is given in Fig. 12. Here, right arm of the source mesh was rotated 90°, and our method produced a plausible deformation.

We also deformed the source mesh by using classical and recent skinning methods to compare them with our method. We generated the same poses given in the top and bottom rows of Fig. 10 by using LBS [2], DQS [20], Fast Automatic Skinning

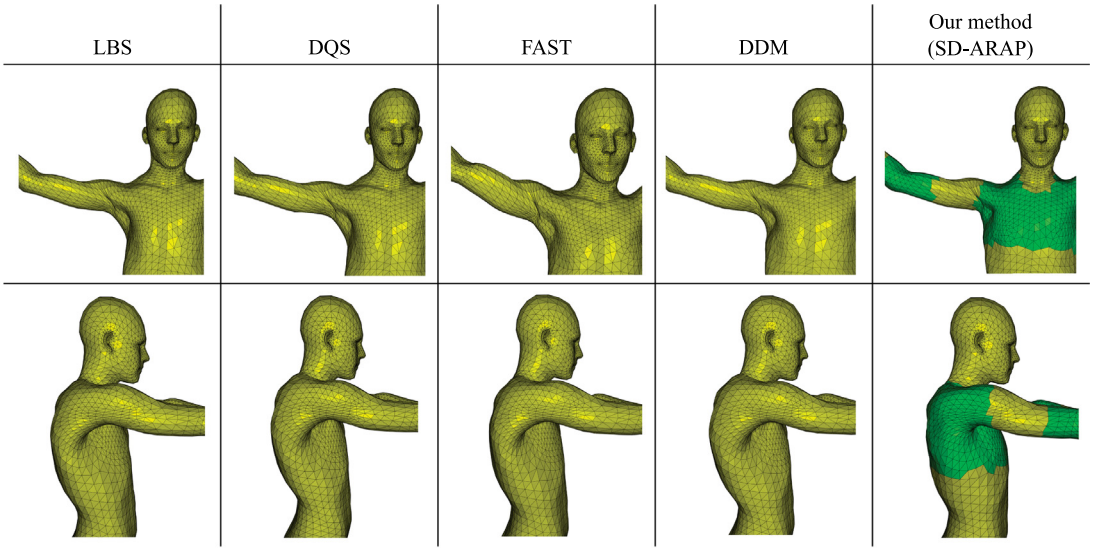


Fig. 13. Same poses shown at top and bottom rows of Fig. 10 are obtained by deforming the source mesh given in Fig. 9 by using different skinning methods. Volume loss artifacts still occur with the skinning methods around the right armpit of the human while the proposed method results in more natural deformations around those areas.

Transformations (FAST) [24], and Direct Delta Mush (DDM) [28]. Aforementioned and well known volume-loss artifacts occurring with LBS and DQS can be noticed around armpits of the human shape in Fig. 13 obtained by deforming the source in Fig. 9. FAST reduces to LBS in our framework without using abstract handles. Thus, we enriched the deformation space by adding abstract handles as suggested. Although this improves the results, the deformation around armpits of the human shape is still unnatural. DDM is a recent skinning method not requiring any blend weight setting yet the results obtained with that are still not as natural as results obtained by our method.

The difference between our method and skinning methods is much more prominent in twisting cases. While twisting with skinning methods causes distortions and singularities around the twisting joint, our method results in much more intuitive and visually appealing results as can be seen in Fig. 14.

It is possible to learn skeleton based deformations by the utilization of existing mesh-skeleton datasets. Neural Blend Shapes (NBS) [32] is a recent example of such method. They trained a neural network by using the data in SMPL [41] and Multi Garment Network [42]. Similar to our problem formulation, the provided source mesh is deformed towards the target skeleton, obtaining the deformed shape. The resulting deformed meshes of two different characters for three poses are depicted in Fig. 15. The qualities of both deformations are parallel yet our method avoids intense training efforts required by supervised learning methods (Section 2.2). Note that such a training should be performed separately for each shape class whereas our method runs without any modifications on a diverse set of classes, e.g., horse, wolf, box, and cylinder. Also, the source shape in NBS should be in T-pose which limits the datasets we can use with it, e.g., MPI FAUST does not have a shape in T-pose so we cannot use NBS on that dataset. Our method does not have such constraints which makes it easier to use.

We tried utilizing the original ARAP and a modified version of it, Smooth-Rotation enhanced ARAP [43], instead of our augmentation process while deforming the source meshes. The resulting meshes along with a brief discussion are provided in the supplementary document. The readers are encouraged to read the discussion to understand better how the augmentation process improves the results that ARAP framework can generate.

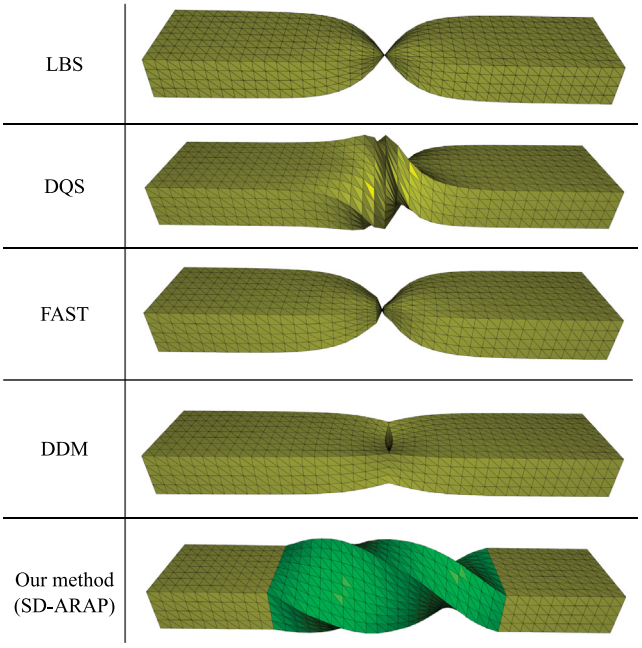


Fig. 14. The box shown in Fig. 11 are twisted 180° by using different skinning methods. The proposed method is significantly more successful than skinning methods to handle twisting near joints without requiring any blend weight painting and parameter tuning.

Quantitative analysis. We quantitatively assessed how well SD-ARAP deformation can approximate to the ground truth deformations in addition to volume loss. To do this, we used the human figure in Fig. 9 as the source mesh, and tried to obtain other poses in MPI FAUST [40] dataset, which were obtained by using real subject. To compute the target skeletons best describing the poses, we first transferred the skeleton samples of the embedded skeleton into the target pose by using registration between the two poses, and constrained them to lie on linear segments so that we can obtain the target skeleton by combining these samples. We transferred the free vertices of the source mesh to new pose with SD-ARAP by moving handles of the source mesh to their

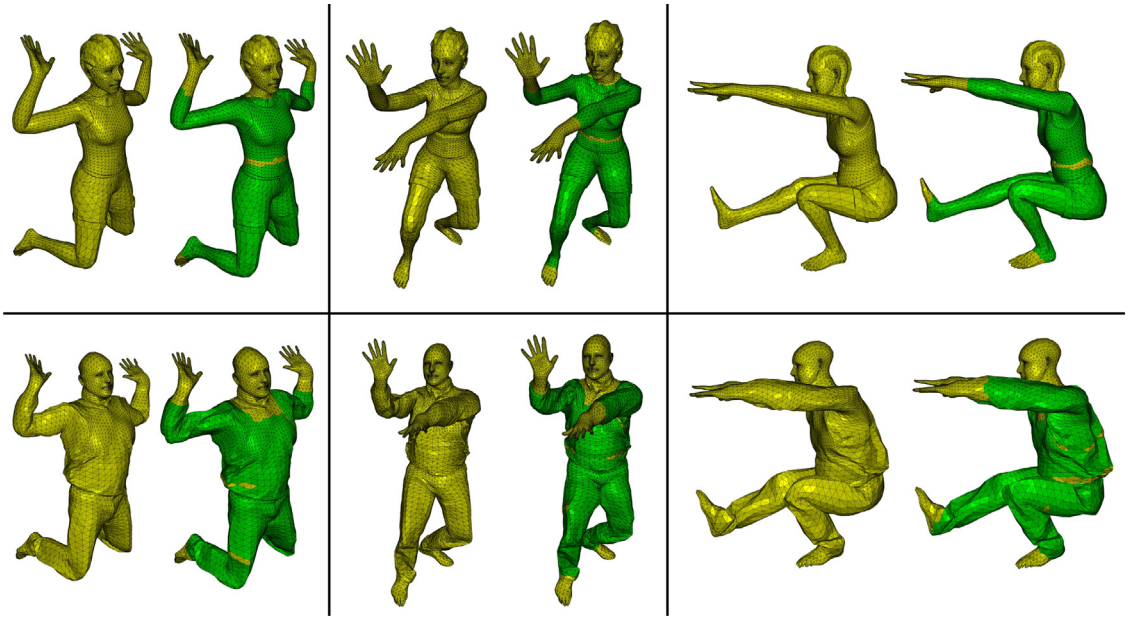


Fig. 15. Comparison of our method with NBS [32]. In each subfigure, the mesh on the left is resulted from NBS while the one on the right is resulted from our approach. The character on the top is from SMPL dataset [41], and the character on the bottom is from Multi Garment Network dataset [42].

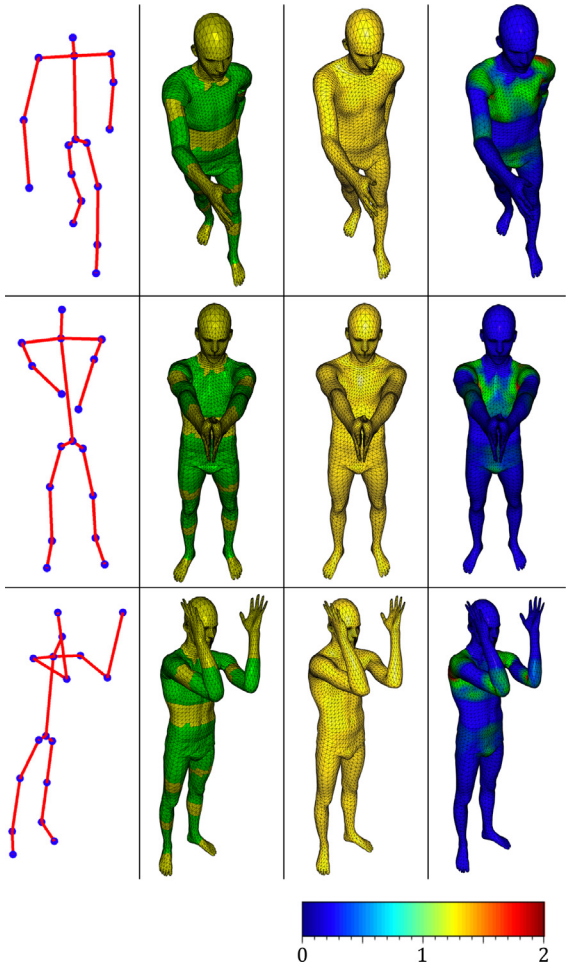


Fig. 16. Reconstructing some poses in the FAUST dataset. The source mesh is shown in Fig. 9. Target skeletons are given in the first and transferred meshes in the second column. Ground-truth poses from the dataset are shown in the third column. Displacement errors $\Delta_i/l_{\mathcal{N}(i)}$ between the approximated and ground-truth mesh vertices are shown in the last column as a color code.

registered vertices on the ground-truth pose and by moving the embedded skeleton to the computed target skeleton. We measured the distance Δ_i between each vertex of the deformed and ground-truth poses. Then we scaled each distance with $1/l_{\mathcal{N}(i)}$ where $l_{\mathcal{N}(i)}$ is the average length of the edges between ith vertex and the vertices in its 1-ring neighborhood. The results of this experiment are stated in Fig. 16 where the target skeleton, transferred mesh, ground-truth mesh, and $\Delta_i/l_{\mathcal{N}(i)}$ values as color codes are given in column by column. Each row shows a different pose.

Timings. Average computation times for obtaining new poses are shown in Table 1. The time consumed in preliminary operations before SD-ARAP optimization like mesh augmentation and LU factorization of the system matrix is reported in "pre". column. The time consumed in the optimization process and the time consumed in each iteration are reported in separate columns. More than % 90 of the time is spent in skeleton embedding according to the timing analysis. A more detailed analysis of the timings along with comparisons with other ARAP methods is provided in the supplementary document.

Example applications. In addition to testing the method on manually drawn stick figures, we also tested it on the stick figures which are outputs of other methods. Those experiments reveal potential applications of the method at the same time. CMU MoCap dataset [44] contains large number of kinematic skeletons in the form of stick figures abstracting the poses of the subjects doing certain movements. We transferred two source shapes from FAUST dataset shown at the leftmost column in Fig. 17 to the skeleton of a running subject, whose poses were captured at 120 Hz frame rate, and abstracted as a kinematic skeleton consisting of 27 vertices and 26 edges. The whole movement consists of 100 frames. The results for the frame numbers multiple of tens are shown in Fig. 17. Transferring the source shape to the target skeletons as a batch took around 4.5s in total.

We also tested the method on the skeletons obtained by VIBE [45], a recent 3D pose estimation method. The 3D kinematic skeleton of a shape in 2D video was estimated by using VIBE, consisting of 17 vertices and 16 edges. Then two source shapes from FAUST dataset (which are different from the ones in other tests, and also in different poses) were transferred to those skeletons.

Table 1
Running time analysis of the whole pipeline. Human model can be seen in Fig. 9, wolf model is shown in 5, and the horse model was depicted in the supplementary document. $|V_{cs}|$ and $|V_{trg}|$ represent the number of vertices in the curve-skeleton an in the target skeleton, respectively. The algorithm was run for 500 iterations, giving us the total skeleton embedding time. $|V_{src}|$ and $|V_{trg}|$ represent the number of vertices and edges, respectively, in the source mesh. The reported timings are obtained with a single core of 2.2 GHz Intel® Core™ i7 processor.

Source	$\|V_{cs}\|$	$\|V_{trg}\|$	Per iter. (ms)	Skeleton embedding (s)	$\|V_{src}\|$	$\|E_{src}\|$	Pre. (ms)	Per iter. (ms)	SD-ARAP opt. (ms)	Total (s)
Human	522	17	36	18.127	6 890	20 664	234	5.25	42	18.403
Horse	654	24	41	20.509	19 248	57 738	1 106	19.21	653	22.268
Wolf	508	19	50.4	25.197	4 344	13 026	25	3.33	100	25.322

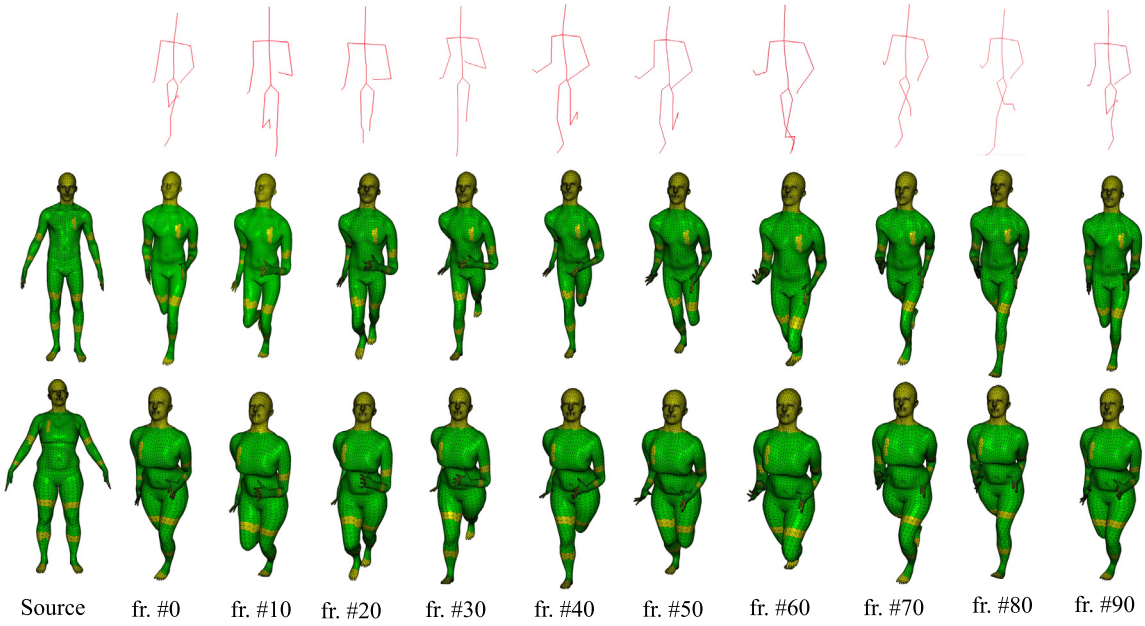


Fig. 17. Mesh transfer results to the kinematic skeletons from the video of a running subject from CMU MoCap dataset (subject #9, trial #1) [44] for 10 frames. The video was captured in 120 Hz. The kinematic skeletons consist of 27 vertices and 26 edges. While the leftmost columns show source shapes, the other ones show transfer results. Each column corresponds to a frame and the frame number is indicated beneath each column.

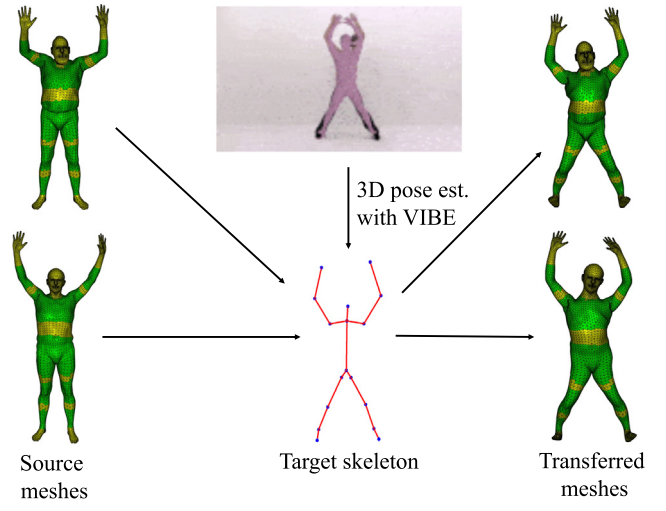
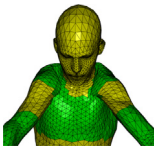


Fig. 18. Mesh transfer results to the kinematic skeletons estimated from a 2D video by using VIBE [45], consisting of 17 vertices and 16 edges. Meshes at left are from FAUST dataset and represent the source meshes. The skeleton in the middle is the estimated pose. The meshes at right are the results of the mesh transfer method.

Results are shown in Fig. 18. Along with those tests, we used the meshes of 4 different human subjects from FAUST dataset.

Limitations. The extracted skeleton and target skeleton matching procedure described in Section 4.2 places the regular vertices from the target skeleton to the embedded skeleton according to Eq. (4). This implies the user should carefully place the regular vertices in the target skeleton. If the user places them far away to articulation regions, the matching procedure has no means to correctly embed them close to articulation regions, which in turn results in unnatural deformations. The figure inset shows a resulting mesh from a target skeleton in which the regular vertices around shoulders are poorly placed. Moreover, our method cannot be used for stretching out limbs of the source mesh as the handles on the mesh surface are transferred rigidly, not non-isometrically, to the target skeleton.



As a limitation in the deformation side, the source mesh should not contain self intersections. If the source has self intersections as in Fig. 19 at left side, support edges cannot be added correctly in the mesh augmentation process, which leads to erroneous transferred meshes as can be seen at the right side of Fig. 19.

7. Conclusion and future work

We proposed a two-stage shape deformation method with which one can (i) repose an articulated character with a 3D stick figure and (ii) transfer a source mesh to this target skeleton. In

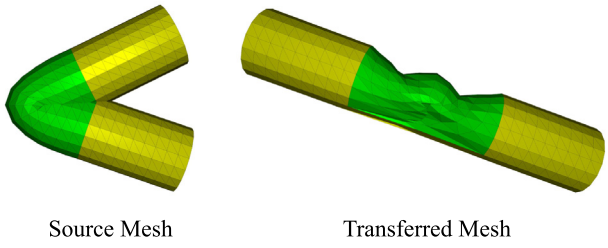


Fig. 19. If the source mesh contains self intersections, it cannot be transferred robustly as the support edges cannot be constructed in the correct way in the mesh augmentation process.

the first stage, the target skeleton is embedded into the source mesh so that the pose of the source mesh is captured by the target skeleton. Then, the source mesh is augmented by introducing new edges between the embedded skeleton and the source mesh. In the second stage, the source mesh is deformed towards the new pose by utilizing the one-to-one correspondence between the embedded skeleton and target skeleton, and by minimizing an ARAP energy defined for the augmented mesh. The first stage can be used for rigging purposes standalone and the second stage can be used for reposing purposes standalone too, if the user already has an embedded skeleton in a source mesh.

The current version of the method does not consider possible self-intersections of the deformed mesh. One of the future work directions is introducing a contact model, so that we can get rid of self-intersections with bulging effect in a realistic manner. Another direction is designing a user-friendly GUI with which one can easily draw target skeletons in 3D, or potentially in 2D, and see the transferred mesh in real-time, which also helps the user to place the regular vertices more easily preventing potential limb stretching. Non-isometry support that would allow additional stretching and squeezing deformations can also be incorporated into our framework.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by TUBITAK, Turkey under the project EEEAG-119E572.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cad.2022.103352.

References

[1] Seylan Ç, Sahillioğlu Y. 3D skeleton transfer for meshes and clouds. Graph Models 2019;105:101041.
[2] Thalmann MN, Laperrière R, Thalmann D. Joint-dependent local deformations for hand animation and object grasping. In: Proc. on graph. interface '88. 1989, p. 26–33.
[3] Sorkine O, Alexa M. As-rigid-as-possible surface modeling. In: Proc. of the fifth eurographics sym. on geom. proc.. 2007, p. 109–16.
[4] Bessmeltsev M, Vining N, Sheffer A. Gesture3D: Posing 3D characters via gesture drawings. ACM Trans Graph 2016;35(6).
[5] Kim VG, Chaudhuri S, Guibas L, Funkhouser T. Shape2pose: Human-centric shape analysis. ACM Trans Graph 2014;33(4).
[6] Aujay G, Hétroy F, Lazarus F, Depraz C. Harmonic skeleton for realistic character animation. In: Proc. of the 2007 ACM SIGGRAPH/Eurographics sym. on comp. anim.. 2007, p. 151–60.
[7] McLaughlin CK, Edelsbrunner H, Harer J, Natarajan V, Pascucci V. Loops in reeb graphs of 2-manifolds. In: Proc. of the nineteenth annual sym. on comp. geometry. Association for Computing Machinery; 2003, p. 344–50.
[8] Xu Z, Zhou Y, Kalogerakis E, Chris L, Landreth C, Singh K. RigNet: Neural rigging for articulated characters. ACM Trans Graph 2020;39(4).
[9] Baran I, Popović J. Automatic rigging and animation of 3D characters. ACM Trans Graph 2007;26(3).
[10] Poirier M, Paquette E. Rig retargeting for 3D animation. In: Proc. of graph. interface 2009. 2009, p. 103–10.
[11] Pascucci V, Scorzelli G, Bremer P, Mascarenhas A. Robust on-line computation of reeb graphs: Simplicity and speed. ACM Trans Graph 2007;26(3):58–es.
[12] Pantuwong N, Sugimoto M. A novel template-based automatic rigging algorithm. Comp Anim Virtual Worlds 2012;23(2):125–41.
[13] Pantuwong N, Sugimoto M. Skeleton-growing: A vector-field-based 3D curve-skeleton extraction algorithm. In: ACM SIGGRAPH ASIA 2010 sketches. 2010.
[14] Luo S, Feng J. Symmetry-aware kinematic skeleton generation of a 3D human body model. Multim Tools Appl. 2020.
[15] Au O, Tai C, Chu H, Cohen-Or D, Lee T. Skeleton extraction by mesh contraction. ACM Trans Graph 2008;27(3):1–10.
[16] Chen J, Izadi S, Fitzgibbon A. Kinêtre: animating the world with the human body. In: Proc. of the 25th Annual ACM sym. on user interface software and technology. 2012, p. 435–44.
[17] Akman A, Sahillioğlu Y, Sezgin M. Generation of 3D human models and animations using simple sketches. In: Proc. of graph. interface. 2020.
[18] Alexa M. Linear combination of transformations. ACM Trans Graph 2002;21(3):380–7.
[19] Kavan L, Žára J. Spherical blend skinning: A real-time deformation of articulated models. In: Proc. of the 2005 Sym. on interactive 3D graphics and games. I3D '05, 2005, p. 9–16.
[20] Kavan L, Collins S, Zára J, O'Sullivan C. Geometric skinning with approximate dual quaternion blending. ACM Trans Graph 2008;27(4).
[21] Rohmer D, Hahmann S, Cani M-P. Exact volume preserving skinning with shape control. In: Proc. of the 2009 ACM SIGGRAPH/Eurographics sym. on comp. anim.. SCA '09, 2009, p. 83–92.
[22] McAdams A, Zhu Y, Selle A, Empey M, Tamstorf R, Teran J, et al. Efficient elasticity for character skinning with contact and collisions. ACM Trans Graph 2011;30(4).
[23] Kavan L, Sorkine O. Elasticity-inspired deformers for character articulation. ACM Trans Graph 2012;31(6).
[24] Jacobson A, Baran I, Kavan L, Popović J, Sorkine O. Fast automatic skinning transformations. ACM Trans Graph 2012;31(4).
[25] Le BH, Hodgins JK. Real-time skeletal skinning with optimized centers of rotation. ACM Trans Graph 2016;35(4).
[26] Vaillant R, Barthe L, Guennebaud G, Cani M, Rohmer D, Wyvill B, et al. Implicit skinning: Real-time skin deformation with contact modeling. ACM Trans Graph 2013;32(4).
[27] Vaillant R, Guennebaud G, Barthe L, Wyvill B, Cani M. Robust iso-surface tracking for interactive character skinning. ACM Trans Graph 2014;33(6).
[28] Le BH, Lewis JP. Direct delta mush skinning and variants. ACM Trans Graph 2019;38(4).
[29] Gao L, Yang J, Qiao Y-L, Lai Y-K, Rosin PL, Xu W, et al. Automatic unpaired shape deformation transfer. ACM Trans Graph 2018;37(6).
[30] Tan Q, Gao L, Lai Y-K, Yanj J, Xia S. Mesh-based autoencoders for localized deformation component analysis. In: Proc. of the AAAI conf. on artificial intelligence, vol. 32. (1). 2018.
[31] Liu L, Zheng Y, Tang D, Yuan Y, Fan C, Zhou K. NeuroSkinning: Automatic skin binding for production characters with deep graph networks. ACM Trans Graph 2019;38(4).
[32] Li P, Aberman K, Hanocka R, Liu L, Sorkine-Hornung O, Chen B. Learning skeletal articulations with neural blend shapes. ACM Trans Graph 2021;40(4).
[33] Sumner RW, Popović J. Deformation transfer for triangle meshes. ACM Trans Graph 2004;23(3):399–405.
[34] Yang J, Gao L, Lai Y-K, Rosin PL, Xia S. Biharmonic deformation transfer with automatic key point selection. Graph Models 2018;98.
[35] Baran I, Vlasic D, Grinspun E, Popović J. Semantic deformation transfer. ACM Trans Graph 2009;28(3).
[36] Roberts RA, dos Anjos RK, Maejima A, Anjyo K. Deformation transfer survey. Comput Graph 2021;94:52–61.
[37] Sahillioğlu Y. A genetic isometric shape correspondence algorithm with adaptive sampling. ACM Trans Graph 2018;37(5):175.
[38] Goldberg D, Lingle R. Alleles, loci, and the traveling salesman problem. In: Proc. of the 1st int. conf. on genetic algorithms. 1985, p. 154–9.
[39] Sorkine-Hornung O, Rabinovich M. Least-squares rigid motion using SVD. 2014, Department of Computer Science, ETH Zurich.
[40] Bogo F, Romero J, Loper M, Black MJ. FAUST: Dataset and evaluation for 3D mesh registration. CVPR 2014;3794–801.
[41] Loper M, Mahmood N, Romero J, Pons-Moll G, Black MJ. SMPL: A skinned multi-person linear model. ACM Trans Graph 2015;34(6).

[42] Bhatnagar BL, Tiwari G, Theobalt C, Pons-Moll G. Multi-garment net: Learning to dress 3D people from images. In: IEEE Int. conf. on comp. vis.. ICCV, 2019.
[43] Levi Z, Gotsman C. Smooth rotation enhanced as-rigid-as-possible mesh animation. IEEE Trans Vis Comput Graph 2015;21(2):264–77.
[44] CMU Graphics Lab. CMU graphics lab motion capture database. 2003, URL http://mocap.cs.cmu.edu/. (Accessed 04 July 2021).
[45] Kocabas M, Athanasiou N, Black MJ. VIBE: Video inference for human body pose and shape estimation. In: CVPR. 2020.