Vis Comput (2018) 34:1285–1297
https://doi.org/10.1007/s00371-017-1413-6

ORIGINAL ARTICLE

Automatic skinning and weight retargeting of articulated characters using extended position-based dynamics

Junjun Pan[1] · **Lijuan Chen**[1] · **Yuhan Yang**[1] · **Hong Qin**[2]

Published online: 21 June 2017

Abstract Animating an articulated character requires the explicit specification of interior skeleton structure and its attachment to skin surface. This task of "rigging" typically involves the manual weight painting and deformation fine-tuning with popular conventional animation methods. Weight painting is unavoidably a time-consuming and laborious process that would need sophisticated skills from animators during animation production. In this paper, using the extended position-based dynamics (PBD), we have articulated a strategy to generate the realistic skin deformation and reuse the painted weights on a new character. For each frame, the skin is deformed by the linear blend skinning method (LBS) at first. To solve the problem of candy-wrapper effect and surface overlapping in LBS, we improve the traditional PBD method by adding energy constraints and employ several geometrically and physically-based constraints to refine the deformed skin automatically. To further reduce the animator's workload from tedious rigging process, we propose the weight retargeting approach using surface matching and interpolation based on the powerful bi-harmonic distance. It could transfer the weights of an existing model to a new character with similar topology. Through numerous experiments, we could demonstrate the visual performance of our new techniques on a variety of articulated characters.

✉ Junjun Pan
pan_junjun@hotmail.com; junjun.pan@gmail.com

Hong Qin
qin@cs.stonybrook.edu

[1] State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China

[2] Department of Computer Science, Stony Brook University (SUNY Stony Brook), Stony Brook, NY, USA

Keywords Linear blend skinning · Position-based dynamics · Weight retargeting · Bi-harmonic distance

1 Introduction

To animate a 3D character, animators need to embed the skeleton into the character model and attach it to the surface mesh. This process is called rigging. Once a character is properly rigged and the influence of the skeletal joints is associated with the skin surface, the task of "skinning" is finished. Such type of animation is called skeleton-driven animation which has become an industrial standard for character animation. Despite many recent research efforts, using the state-of-the-art 3D software such as Maya and 3dmax, manual rigging is still a common way for most animation studios. Building quality rigs would require animators spending a significant amount of time and effort to manipulate the models, especially during the process of weight painting.

Traditional rigging process of articulated characters could be roughly decomposed into the following tasks:

1. *Skeleton embedding* Setting a hierarchy of skeleton structure for the character; adjusting the position and orientation of joints in the skeleton to fit the model properly.
2. *Skinning* Binding the skeleton to the model; influencing the character surface mesh with a skinning algorithm and refining plenty of attributes, such as weight, in order to obtain proper deformation.
3. *Setting controllers* Creating handles and manipulators to control the rig, by a series of constraints or inverse kinematics (IK).

Many research institutions and studios have made great efforts to reuse and automate the above-mentioned labor-intensive process for different models. In principle, using a

template skeleton structure with a fixed number of bones and joints, tasks (1) and (3) could be conducted automatically [1–3]. However, the automation of task (2) is difficult since the rigging process is strongly dependent on the shape of the character model, which varies wildly from one to the others. Hence, the skinning process in professional studios usually requires a large amount of back and forth manual modification of the skin weights (i.e., weight painting) by highly skilled animators, in order to ensure a realistic and smooth surface deformation. Although there have been a number of studies [4–6] to address this very important topic, none of them has satisfactorily achieved the goal of generating high-quality rig without manual interventions.

In this paper, we develop an automatic skinning and weight retargeting approach for articulated characters. After a template skeleton embedding, the skin is deformed by the linear blend skinning method at first. To solve the problem of "candy-wrapper" and surface overlapping in LBS, we improve the traditional position-based dynamics method by adding energy constraints and employ a number of geometric and physical constraints to modify the deformed skin in joint area automatically. Since most weight painting takes place at joint areas, this PBD-based skinning refinement could save animators a significant amount of time from the traditional rigging process. Finally, to further release the animator from tedious weight painting, we devise the weight retargeting algorithm using the surface matching and interpolation method based on the bi-harmonic distance. It can transfer the well-painted weight from an existing character model to a new one with similar topology. This process ensures the high-quality rig can be reused without manual interventions. Specifically, this paper includes the following innovative contributions:

- We extend the traditional PBD method by adding energy constraints. Such constraints could enforce energy preservation, which takes the stiffness of material into account and can illustrate the visual performance of soft skin deformation with different material attributes. Then, we employ a number of geometric and physical constraints (e.g., stretch, volume preservation, bind, energy preservation, self-collision) in PBD to modify the deformed skin by LBS in joint area. This automatic skinning refinement process can reduce animators a significant amount of time during the process of weight painting.
- We propose a weight retargeting approach using the surface matching and interpolation method based on the bi-harmonic distance. It can transfer the well-painted weights from an existing character model to a new one with similar topology. Coupled with the automatic skinning refinement based on PBD, this technique ensures the high-quality rig can be reused without manual interventions.

2 Related work

Our method is closely related to skeleton-based skinning techniques and position-based dynamics. We list the references that are most relevant to our work.

Skeleton-based skinning techniques can be classified into two categories; geometry-based methods and example-based methods. Geometry-based approaches [7–9,12] deform the articulated characters by the standard skeletal subspace deformation. One well-known algorithm is the linear blend skinning (LBS) [7], which has been widely used in the real-time animation due to its high computational efficiency. However, LBS suffers from several visual distortion, such as self-intersection and "candy-wrapper" artifact, because of the linear nature of the algorithm. Through a nonlinear transformation blending, e.g., dual quaternion skinning (DQS) [10], the geometry distortion of LBS can be reduced. Unfortunately, DQS blending also suffers from the undesirable joint-bulging artefacts, which requires manual work from animators to be fixed [11]. Since all the manual work in rigging, such as weight painting and artefacts fixing, are time-consuming process, automatic skinning techniques [4,5] are becoming increasingly popular. Jacobson et al. proposed a linear blending weights, called bounded bi-harmonic weights, to produce smooth and intuitive deformations for points, bones and cages of arbitrary topology [5]. These weights minimize the Laplacian energy subject to bound constraints. It can be used for real-time deformation of 2D and 3D shapes. Mukai et al. propose a method for dynamic skinning with a helper bone rig [9]. It can procedurally control helper bones according to skeleton motion in order to imitate dynamic skin deformation. Vaillant et al. use implicit surfaces to preserve the original mesh properties, produce controllable deformations and avoid self-collision during deformation [12,13]. Müller et al. enhance the realism of animated characters by adding physically-based secondary motion to deformable parts such as cloth, skin or hair [14].

Example-based methods are another effective way to deform skin. They usually remove the geometry artefacts by adding pose examples [15–18] or additional skinning weights [19,20]. Xu et al. presented an approach to add physically-based dynamics to pose-space deformation and character rigging [18]. It augments pose-space deformation with high-quality secondary soft tissue dynamics under arbitrary rigging. Mohr et al. present an automated method to build character skins that are fast to compute and compactly represented from a set of examples [19]. It allows artists to use any skin editing tools they like while producing characters that meet the visual performance demands.

Weight setting for skinning is also an important task in rigging to avoid deformation artefacts. Recently, Dionne et al. [6] presented an automatic skinning weights computation

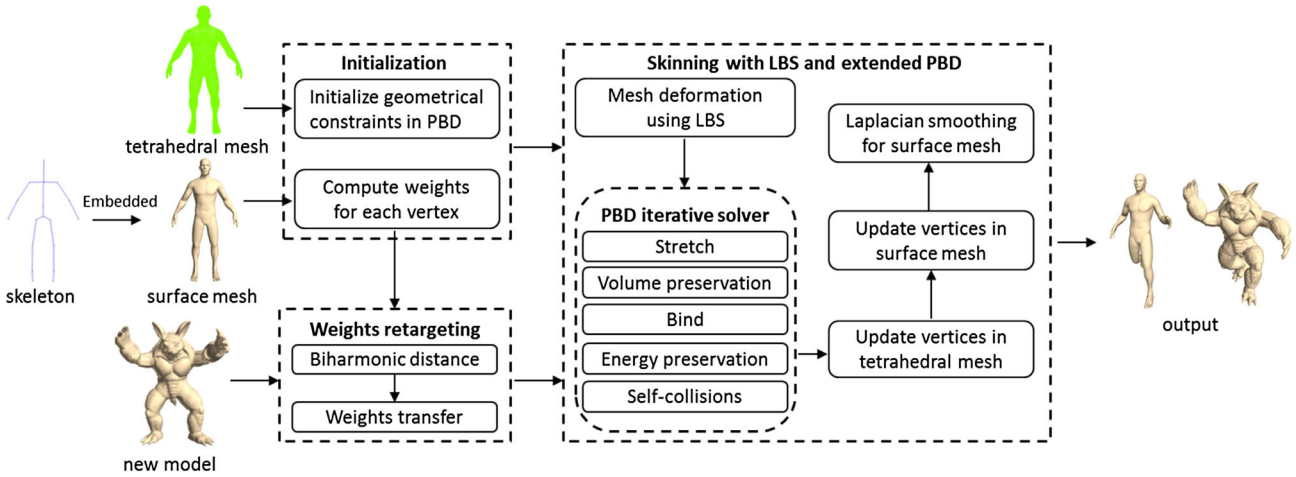


Fig. 1 Framework of automatic skinning and weight retargeting by extended position-based dynamics

algorithm using geodesic distances. Based on the interpolation in the surface and volume domain, Seo et al. present a method to transfer an anatomically-based rig from an existing character to another [21]. Ju et al. presented a skinning template method using cage-based deformations [22]. It allows the rigging and skinning solutions to be shared and reused for new articulated characters. However, this cage-based skinning template approach cannot create detailed skin behaviors which are usually required in high-end applications. Le et al. pre-compute the optimized center of rotation for each vertex from the rest pose and skinning weights [23]. During animation, these centers of rotation are used to interpolate the rigid transformation for each vertex. This method can significantly reduce the artifacts of LBS and DQS.

To take full advantage of painted weights for existing characters, some innovative methods involve weight retargeting. Allen et al. present a method for transferring the animation setup from one character to other scanned character meshes [24]. Each joint is retargeted according to three markers on the meshes. Dicko et al. present a method which can retarget several attributes, including skeleton and skinning weights [25]. Nevertheless, this method cannot retarget user provided skeletons and skinning weights as it relies on a template character. Recently, Avril et al. present a general method for transferring skeletons and weights between characters with distinct mesh topologies [26].

Position-based dynamics (PBD) is a wildly used method in physical deformation, due to its fast, robustness and position-based manipulation feature [27,28]. It has unique advantages in deforming cloth and soft object and becomes increasingly popular in the game industry and VR surgical simulator development [29]. Müller et al. presented a unified PBD framework to simulate rigid, soft objects and fluid in real-time environment [30]. Rumman et al. presented a skinning algorithm for skeleton-driven deformations of articulated characters based on position-based dynamics [31]. The whole process can be divided into two steps. In the first step, the character is deformed by LBS; then, in the second step the position of the vertices is accommodated by a PBD solver. A graph coloring algorithm is also employed for parallelizing the computation of the geometrical constraints. This leads to a fast-deformed skin refinement. However, their approach does not resolve the self-intersection, which leads to geometry overlapping. Our research is the further work based on [31]. We extend PBD with more constraints, which contain the energy preservation and self-collision, to handle the material and overlapping problem. We also apply the surface matching and interpolation based on bi-harmonic distance, in the weight retargeting for rigging reuse.

3 Overview

As shown in Fig. 1, we first present the framework overview as follows:

3.1 Initialization

The input is a triangle mesh representing the surface of character model and an animated skeleton. We use the technique in [4] to embed the skeleton into the triangle mesh. The tetrahedral mesh is also generated as the interior structure of character using the tool package: PhysxViewer in NVIDIA PhysX SDK [32]. We also compute the weights for each vertex.

3.2 Skinning with LBS and extended PBD

At each frame, the skeleton moves and both the surface and volumetric vertices are deformed by a standard LBS algo-

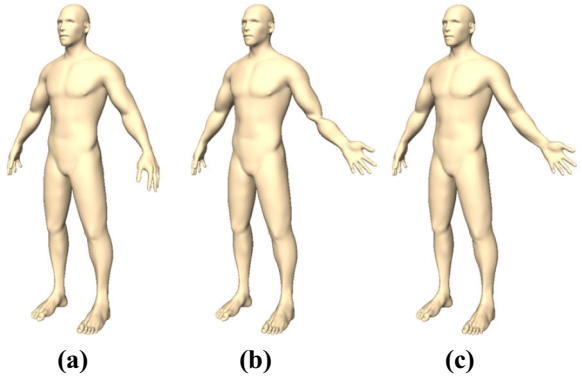


Fig. 2 Different stages of automatic skinning in our approach. **a** Original model, **b** deforming using a standard LBS, **c** refining the surface mesh using extended PBD

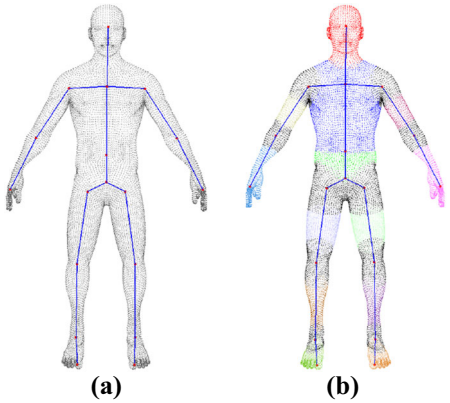


Fig. 3 Decomposition of all the vertices. **a** Original object with embedded skeleton, **b** decomposition result

rithm (Fig. 2b). Then, we use position-based dynamics to update both the tetrahedral and surface mesh automatically in joints area by solving the constraints, which consists of stretch, volume preservation, bind, self-collision and energy preservation (Fig. 2c). Finally, the surface mesh will be output after Laplacian smoothing.

3.3 Weight retargeting

In order to further improve the efficiency of rigging process, a weight retargeting algorithm based on bi-harmonic distance is designed to transfer the well-painted weights from an existing character model to a new one with similar topology. With a well retargeted weights, the visual performance of skinning with LBS and extended PBD can be improved.

4 Automatic skinning based on extended position-based dynamics

4.1 Linear blend skinning

Generally, the animation skeleton can be defined by bone segments connected with joints. In each animation frame, supposing $\mathbf{v}_i$ is the location of a surface vertex at its rest pose, the deformed result $\mathbf{v}_i^{'}$ can be computed by:

$$\mathbf{v}_i^{'} = \sum_{j=1}^{m} w_{i,j} \mathbf{W}_j \mathbf{B}_j^{-1} \mathbf{v}_i, \tag{1}$$

where $\mathbf{W}_j$ denotes the transformation matrix associated with bone j in its current pose. $\mathbf{B}_j^{-1}$ denotes the inverse transformation of bone j in the rest pose. Here $j = 1 \ldots m$ and m is the total number of bone segments. $w_{i,j}$ is the weight binding $\mathbf{v}_i$ to bone j. We can know that $w_{i,1} + \ldots + w_{i,m} = 1$ and $w_{i,j} \geq 0$.

In LBS, each vertex is usually attached to one or more skeletal bones. For each vertex $\mathbf{v}_i$, the weight w_j indicates how strong the bone j can affect on it in deformation. To determine the weights, we first employ a simple but effective method [3] to decompose all the vertices into different regions (Fig. 3b). It is similar to the nearest-neighbor clustering. We first collect the vertices near the joint under a radius threshold, which can be set by experimental experience. All these vertices belong to the joint regions. The rest of vertices will be classified according to its nearest bone segment. If a vertex is in the joint area (black points in Fig. 3b), it is attached to 3 bone segments. If a vertex is not in the joint area, it is attached to 2 bone segments. Then, we compute the weights by:

$$w_{i,j} = \frac{(1/d_{i,j})^4}{\sum_{k=1}^{n_i} (1/d_{i,k})^4}, \tag{2}$$

where $d_{i,k}$ indicates the distance between the vertex $\mathbf{v}_i$ and its k-th attached bone segment, n_i is the total number of bone segments attached to the vertex $\mathbf{v}_i$. The deformation result of LBS is smooth but suffer from several artifacts such as collapse (Fig. 2b) and self-intersection in joints area. We need fix these artifacts by PBD which will be described in the next section.

4.2 Position-based dynamic

Position-based dynamics (PBD) is a real-time deformation approach based on Verlet integration. It is widely used in the physical-based animation with interactions [29]. Here we use PBD to modify the deformed skin in joints area through its geometric and physical constraints. Technically, the initial structure of object is modeled as the tetrahedral mesh with a set of N particles and M constraints. A particle i ($i \in [1, \ldots, N]$) has the attributes of mass m_i, the position $\mathbf{p}_i$ and the velocity $\mathbf{v}_i$. The motion of particles is governed by a set of

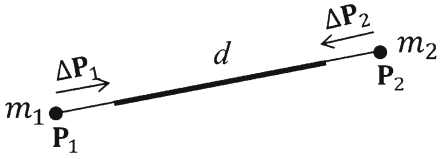


Fig. 4 Illustration of stretch constraint between $\mathbf{p}_1$ and $\mathbf{p}_2$ for a edge

nonlinear constraints C_j ($j \in [1, \ldots, M]$). The system with constraints can be solved by Gauss–Seidel iterations which directly update the position of particles. The target is to find the correction $\triangle\mathbf{p}$ in the neighborhood of constraint function C around the current configuration $\mathbf{p}$ [27], there is:

$$C_j(\mathbf{p} + \triangle\mathbf{p}) \approx C_j(\mathbf{p}) + \nabla_{\mathbf{p}} C_j(\mathbf{p}) \cdot \triangle\mathbf{p} = 0. \tag{3}$$

For the correction $\triangle\mathbf{p}$ of an individual particle $\mathbf{p}_i$, we have

$$\triangle\mathbf{p}_i = -s\nabla_{\mathbf{p}_i} C(\mathbf{p}_1, \ldots, \mathbf{p}_n), \tag{4}$$

$$s = \frac{C(\mathbf{p}_1, \ldots, \mathbf{p}_n)}{\sum_j \left|\nabla_{\mathbf{p}_i} C(\mathbf{p}_1, \ldots, \mathbf{p}_n)\right|^2}. \tag{5}$$

Here we use five types of constraints: stretch, volume preservation, bind, energy preservation and self-collision to modify the deformed skin and fix the artifacts in joints area. Stretch and volume preservation constraints can be found in the traditional PBD technique [27].

a. Stretch constraint of two particles ($\mathbf{p}_1$, $\mathbf{p}_2$) is defined as the elastic edge of triangles in tetrahedral mesh (Fig. 4). It can be described as:

$$C_{stretch}(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d, \tag{6}$$

where d is the rest length of the edge. m_1 and m_2 indicate the mass of particles $\mathbf{p}_1$ and $\mathbf{p}_2$.

b. Volume preservation constraint is defined for the particles ($\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$) at the corners of tetrahedron (Fig 5). It is designed to maintain the initial volume of each tetrahedron and then maintain the total volume of character. The volume preservation constraint can be described as:

$$C_{volume}(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) = \frac{1}{6}(\mathbf{p}_1 - \mathbf{p}_2) \times (\mathbf{p}_1 - \mathbf{p}_3) \cdot (\mathbf{p}_1 - \mathbf{p}_4) - V, \tag{7}$$

where V is the initial volume of the tetrahedron.

c. Bind constraint is a stretch constraint between a particle and its projection on the nearest skeleton bone. We use the method of [31] to define the bind constraint. While moving, the projection point for each particle is updated accordingly

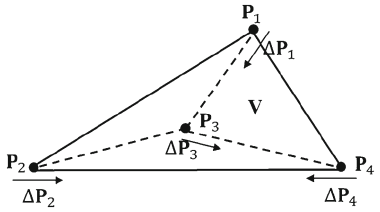


Fig. 5 Illustration of volume preservation constraint

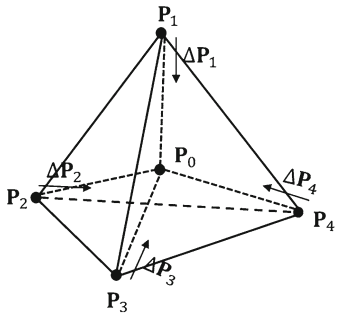


Fig. 6 A energy constraint is defined for a tetrahedron. $\mathbf{p}_0$ is the barycenter of the tetrahedron

and the bind constraint push or pull the particle to maintain the rest distance.

To improve the graphic performance of PBD in skeleton animation, we design another two constraints: energy preservation and self-collision.

d. Energy preservation constraint Currently, none of constraints used in PBD [28] are concerned with material property of object. Here we extend the traditional PBD through defining a energy preservation constraint, which uses the spring potential of each deformed tetrahedron with spring stiffness (elastic coefficient). By setting different spring stiffness, it can give plausible deformation for the character with variety of bio-mechanical material properties.

Figure 6 shows an example of energy preservation constraint. Supposing there are four virtual springs between the barycenter $\mathbf{p}_0$ and four corner particles $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$, there is:

$$\mathbf{p}_0 = \frac{\sum_{i=1}^{4} m_i \mathbf{p}_i}{\sum_{i=1}^{4} m_i}. \tag{8}$$

According to Hooke's law, we use the following equation to express the constraint of energy preservation:

$$C_{energy}(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) = \frac{1}{2}\sum_{i=1}^{4} k_i(|\mathbf{p}_i - \mathbf{p}_0| - d_i)^2, \tag{9}$$

where k_i is the elasticity coefficient of the spring $\mathbf{p}_i\mathbf{p}_0$, d_i is the rest length of the spring $\mathbf{p}_i\mathbf{p}_0$. The gradient with respect to each particle is:

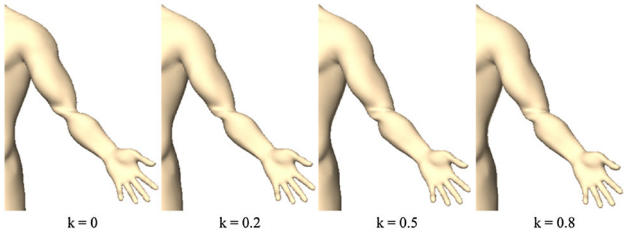


Fig. 7 Twisting the elbow by extended PBD with only energy preservation constraint in different spring stiffness

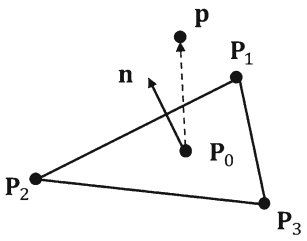


Fig. 8 A self-collision constraint is defined between a particle **p** and a triangle $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$

$$\nabla_{\mathbf{p}_i} C = k_i(|\mathbf{p}_i - \mathbf{p}_0| - d_i)\frac{\mathbf{p}_i - \mathbf{p}_0}{|\mathbf{p}_i - \mathbf{p}_0|}. \tag{10}$$

Then, we can get the correction of each particle:

$$\triangle \mathbf{p}_i = -\frac{w_i C_{energy}(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4)}{\sum_{j=1}^{4} w_j \left|\nabla_{\mathbf{p}_j} C\right|^2}\nabla_{\mathbf{p}_i} C. \tag{11}$$

Figure 7 demonstrates the effect of energy preservation constraint by twisting the elbow (120 degree) with different spring stiffness (9). In this test, only energy preservation constraint is used in the extended PBD. From the figure, the arm shows more "volume preservation" feature when k increases. This constraint can reflect the material attribute of character during deformation.

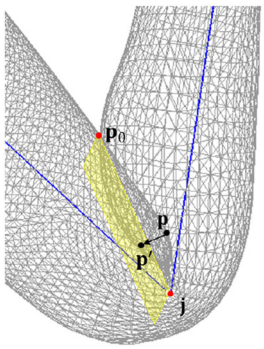
e. Self-collision constraint In skeleton-driven animation, the limbs in joints area may overlap when there is a self-collision. The particle probably moves through a triangle so that self-intersection will occur in the surface mesh. We define the self-collision constraint between a particle and a triangle to maintain the particle on the original side of the triangle (Fig. 8). It can be described as:

$$C(\mathbf{p}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) = (\mathbf{p} - \mathbf{p}_0) \cdot \frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)}{|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)|}, \tag{12}$$

$$C(\mathbf{p}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) \quad C(\mathbf{p}', \mathbf{p_1}', \mathbf{p_2}', \mathbf{p_3}') \geq 0. \tag{13}$$

where $\mathbf{p}_0$ is the barycenter of a triangle $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$, which is the nearest triangle for the vertex **p**. $\mathbf{p_i}'$ indicates the position of $\mathbf{p_i}$ at next animation frame. All the vertices belong to the joints area.

Fig. 9 When self-collision happens, the vertex **p** is projected to $\mathbf{p}'$ which is on the projection plane (*yellow*)



During animation, once Eq. (13) is not met, a self-collision is regarded happened. We employed a straightforward method similar to the "vertex projection" and "tangential relaxation" in [12] to handle the overlapping. Figure 9 illustrates the vertex projection process. The yellow plane indicates the projection plane, which is perpendicular to the plane of bone segments (blue lines). $\mathbf{p}_0$ is the farthest contact vertex from the joint **j**. Both $\mathbf{p}_0$ and **j** are on the projection plane. We project all the contact vertices on the projection plane. Supposing **p** is the contact vertex and **n** is the normal vector of projection plane, the projected contact vertex $\mathbf{p}'$ can be computed as follows:

$$\mathbf{p}' = \mathbf{p} - (\mathbf{n} \cdot (\mathbf{p} - \mathbf{j}))\mathbf{n} \tag{14}$$

Simply using vertex projection method could introduce high distortions of geometry, at the extreme self-intersections cases [12]. In order to minimize the distortions, we introduce the tangential relaxation steps, which move each vertex toward the weighted centroid of its neighbors. Supposing $\mathbf{p}_i$ is the projected vertex, $\mathbf{q}_{i,j}$ are its one-ring neighboring vertices. We compute the barycentric coordinates $\Phi_{i,j}$ such that $\mathbf{p}_i = \Sigma\Phi_{i,j}\mathbf{q}_{i,j}$, using the mean value coordinates. The tangential relaxation which moves vertex $\mathbf{p}_i$ can be computed by

$$\mathbf{p}_i \leftarrow (\mathbf{p}_i + \Sigma\Phi_{i,j}\mathbf{q}_{i,j})/2. \tag{15}$$

Figure 10 illustrates the difference of bending the leg with and without self-collision constraint.

4.3 Surface smoothing

After the modification of deformed skin by PBD, in order to output realistic skin deformations, we apply the Laplacian smoothing to the final surface mesh in joints area. The Laplacian smoothing can be described as following operation:

$$\mathbf{p}_i = (1 - k_i)\mathbf{p}_i + k_i\tilde{\mathbf{p}}_i. \tag{16}$$

where $\tilde{\mathbf{p}}_i$ is the centroid of the one-ring neighborhood vertex of $\mathbf{p}_i$, and k_i controls the amount of smoothing. This opera-

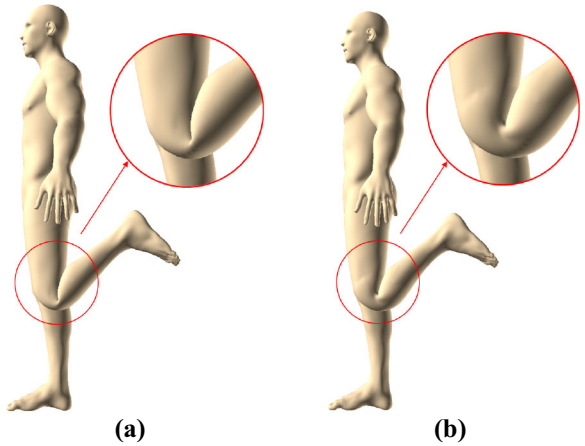


Fig. 10 Bending the leg with and without self-collision constraint. **a** Without the constraint, **b** with the constraint

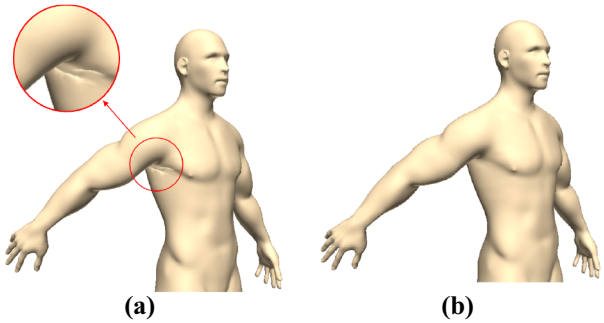


Fig. 11 Change of skin in armpit area for a human model after manually weight painting. **a** Before manually weight painting, **b** after manually weight painting

tion is only done for surface mesh vertices in joints area, so that surface details can be preserved as well.

Algorithm 1 describes the whole process of our skinning method:

5 Weight retargeting

For our technique, though extended PBD can refine the deformed skin and fix the artifacts automatically, large degree deformation still generates non-smooth skin surface occasionally (Fig. 11) if the weight setting is not correct in LBS. Hence to perform a perfect rig for the character, the accurate weight painting by animator's hands is still necessary.

To further release the animator from tedious weight painting which requires sophisticated skills, we also design the weight retargeting technique using the surface matching and interpolation based on bi-harmonic distance. It can transfer the well-painted weight from an existing character model to a new one.

Algorithm 1 Automatic skinning based on extended PBD

Require: A triangle mesh and a skeleton
Ensure: The deformation of triangle mesh
1: embed skeleton into the triangle mesh
2: generate the tetrahedral mesh from triangle mesh using PhysxViewer
3: initialize constraints in PBD
4: compute weights for each vertex
5: surface mesh deformation using LBS
6: **loop** $solveriterations\ times$
7: stretching constraints($\mathbf{c}_{s1},\ldots,\mathbf{c}_{sn}$)
8: volume preservation constraints($\mathbf{c}_{v1},\ldots,\mathbf{c}_{vn}$)
9: bind preservation constraints($\mathbf{c}_{b1},\ldots,\mathbf{c}_{bn}$)
10: energy preservation constraints($\mathbf{c}_{e1},\ldots,\mathbf{c}_{en}$)
11: self-collisions constraints($\mathbf{c}_{c1},\ldots,\mathbf{c}_{cn}$)
12: **end loop**
13: update vertices in tetrahedral mesh
14: update vertices in triangle mesh
15: Laplacian smoothing

5.1 Bi-harmonic distance

Based on the Green's function of the Bi-Laplacian, bi-harmonic distance is a smooth, locally isotropic metric, which has a number of significant advantages, such as globally shape-awareness, isometry-invariance, insensitivity to noise and practical to computation [33]. Moreover, it does not depend on any parameters. The formal definition of bi-harmonic distance operator can be described in few equivalent ways. Here we use the eigenvectors and eigenvalues of the Laplace-Beltrami operator to define the square of the distance:

$$d_B(x,y)^2 = \sum_{k=1}^{\infty} \frac{(\varphi_k(x) - \varphi_k(y))^2}{\lambda_k^2}, \tag{17}$$

where $\varphi_k(x)$, λ_k are the eigenfunctions and eigenvalues and the technical detail can be found in [34]. In particular, we know the Green's function $g_B(x,y)$ of the bi-harmonic operator $\triangle^2$ is

$$g_B(x,y) = \sum_{k=1}^{\infty} \frac{\varphi_k(x)\varphi_k(y)}{\lambda_k^2}. \tag{18}$$

So Eq.17 can be written in the following way:

$$\begin{aligned} d_B(x,y)^2 &= \sum_{k=1}^{\infty} \frac{|\varphi_k(x)|^2}{\lambda_k^2} + \sum_{k=1}^{\infty} \frac{|\varphi_k(y)|^2}{\lambda_k^2} \\ &\quad -2\sum_{k=1}^{\infty} \frac{\varphi_k(x)\varphi_k(y)}{\lambda_k^2} \\ &= g_B(x,x) + g_B(y,y) - 2g_B(x,y). \end{aligned} \tag{19}$$

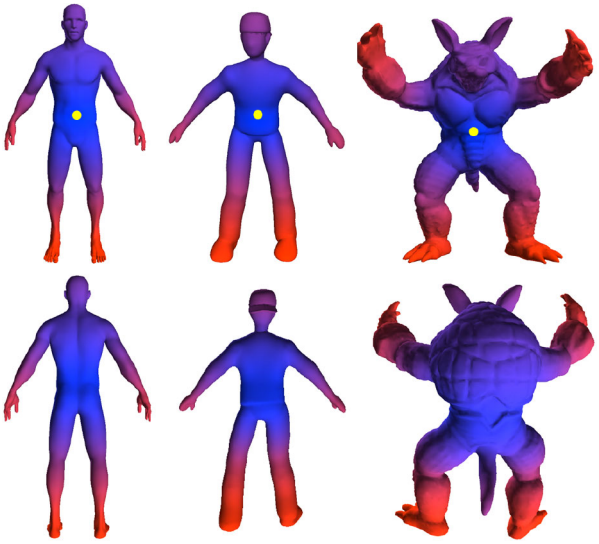


Fig. 12 Bi-harmonic distance distribution of three models from a source point (the *yellow point*). From left to right: human, doll and armadillo

Given a set of vertices in our character model, we are interested in calculating all-pairs of distances in the set. Hence we can get the bi-harmonic distance between any pair of vertices quickly ($O(1)$).

Figure 12 illustrates the bi-harmonic distance distribution of three models (human, doll and armadillo) from a source point at belly button. The dark blue indicates the areas which are nearest from the source point, and the dark red indicates the areas are furthest from the source point.

5.2 Surface matching

Before weight retargeting, we ensure the original model and retargeted model have the same topology of skeleton structure. Then, these two models should be normalized into the same size. We suppose that the surface mesh of original model is N and the surface mesh of new model is M. Firstly, we give a set of corresponding pairs $(x_i, y_i) \in M \times N, i = 1, \ldots, k$, and k is the total number of the corresponding pairs, x_i is a vertex in the M, y_i is a vertex in N. In our experiments, we set k as 5 and the corresponding pairs are selected in the regions of head, hands and knees (Fig. 13). Now given a vertex $x \in M$, we represent it using the vector of distances to the known corresponding points on M , the function is as follows:

$$\mathbf{x} = \frac{1}{max_i d(x, x_i)} (d(x, x_1), d(x, x_2), \ldots, d(x, x_k)), \quad (20)$$

where $d(x, x_i)$ is the bi-harmonic distance between vertices x and x_i. And the normalization is conducted to be more robust to scales between M and N. Similarly for each vertex $y \in N$, it can be represent as:

$$\mathbf{y} = \frac{1}{max_i d(y, y_i)} (d(y, y_1), d(y, y_2), \ldots, d(y, y_k)). \quad (21)$$

In order to find the corresponding point $y \in N$ for each point $x \in M$, we use the spirit of [33] to define a prediction function

$$c(y, x) = |\mathbf{x} - \mathbf{y}|_w = \left(\sum_{i=1}^{k} w_i(x) \, |d(x, x_i) - d(y, y_i)|^2 \right)^{\frac{1}{2}}. \quad (22)$$

where $c(y, x)$ reflects the difference of bi-harmonic distance between corresponding points pair in original model and retargeting model. $w_i(x) > 0$ and $w_i(x)$ indicate the weight for the bi-harmonic distance difference between a points pair in surface matching. Considering the global feature of bi-harmonic distance, to make the computation efficient and straightforward, we treat $w_i(x) = 1$ in our experiment.

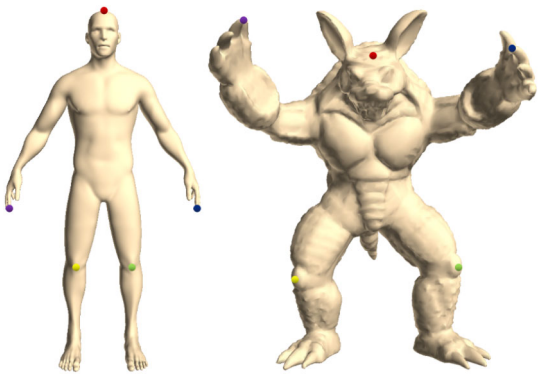


Fig. 13 Five pairs of corresponding points input are selected in the regions of head, hands, knees of the original model (*left*) and retargeting model (*right*) in our experiment

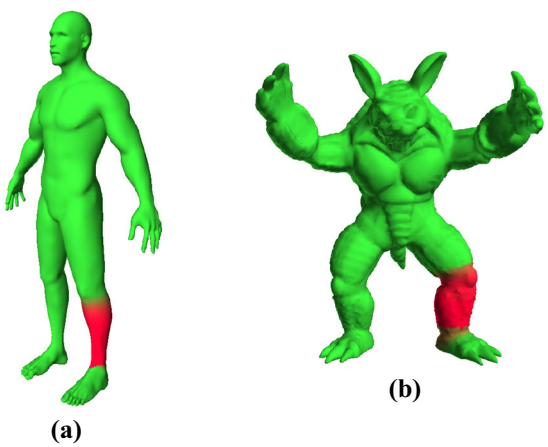


Fig. 14 Weight retargeting for the lower leg region from a human to an armadillo. **a** Original model (*human*), **b** retargeting model (*armadillo*)

We search the vertex y which makes the prediction function (22) minimum to be the corresponding vertex $x^{'}$. To reduce the computation cost, the searching is only limited in the corresponding decomposed region according to the decomposition result (Fig. 3). We also need to find other four points: $y_1, y_2, y_3, y_4 \in N$, which make the prediction function relatively small. To keep a balance of interpolation accuracy and computation cost, we choose the number of interpolation points as four by experimental experience and compute the weight of $x^{'}$ using Shepard interpolation [35]. The Shepard interpolant can be defined as:

$$\mathbf{w}_{x'} = \frac{\sum_{i=1}^{4} f_i(x^{'})\mathbf{w}_i}{\sum_{i=1}^{4} f_i(x^{'})}, \tag{23}$$

where $\mathbf{w}_i, i = 1, \ldots, 4$ is the weight vector of y_i, and $f_i(x^{'})$ can be given by $f_i(x^{'}) = 1/d(y_i, x^{'})$. Here d is the bi-harmonic distance.

Algorithm 2 describes the whole process of weight retargeting. Figure 14 illustrates the weight retargeting result of the lower leg from a human model to an armadillo model based on bi-harmonic distance.

Algorithm 2 Weight retargeting

1: $N \leftarrow model_{original}, M \leftarrow model_{new}$
2: compute Bi-harmonic distance for N,M
3: select correspondence vertices $(x_i, y_i) \in M \times N$
4: **for** x $\in M$ **do**
5: $\quad \mathbf{x} \leftarrow \frac{1}{max_i d(x,x_i)}(d(x,x_1), d(x,x_2), \ldots, d(x,x_k))$
6: **end for**
7: **for** y $\in N$ **do**
8: $\quad \mathbf{y} \leftarrow \frac{1}{max_i d(y,y_i)}(d(y,y_1), d(y,y_2), \ldots, d(y,y_k))$
9: **end for**
10: **for** x $\in M$ **do**
11: $\quad$ **for** i $\in$ [1 *to* 4] **do**
12: $\quad\quad$ minimize predict function $c(y,x) \leftarrow |\mathbf{x}-\mathbf{y}|_w = (\sum_{j=1}^{k} w_j(x)\left|d(x,x_j) - d(y,y_j)\right|^2)^{\frac{1}{2}}$
13: $\quad$ **end for**
14: $\quad$ compute weight for x using Shepard interpolation $\mathbf{w}_{x'} \leftarrow \frac{\sum_{i=1}^{4} f_i(x^{'})\mathbf{w}_i}{\sum_{i=1}^{4} f_i(x^{'})}$
15: **end for**

6 Experiments and comparison

We have implemented our automatic skinning and weight retargeting technique using C++ and OpenGL. We also employ the graph coloring-based parallel PBD technique in [31] and apply CUDA for computation acceleration. All the experiments run on a desktop with NVIDIA GeForceGT 630, Intel(R) Core(TM) i7-4790 CPU (3.60GHz, 8 cores), and 8G RAM. To verify the effectiveness of our method, we have designed two groups of experiments.

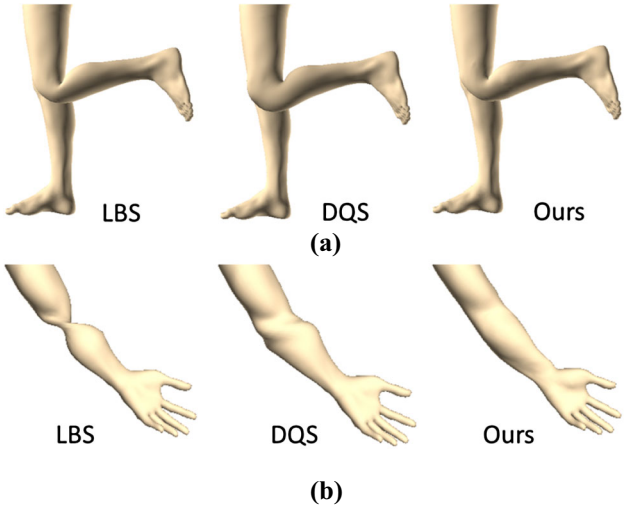


Fig. 15 Comparison of our method with the LBS and DQS. **a** Bending, **b** twisting

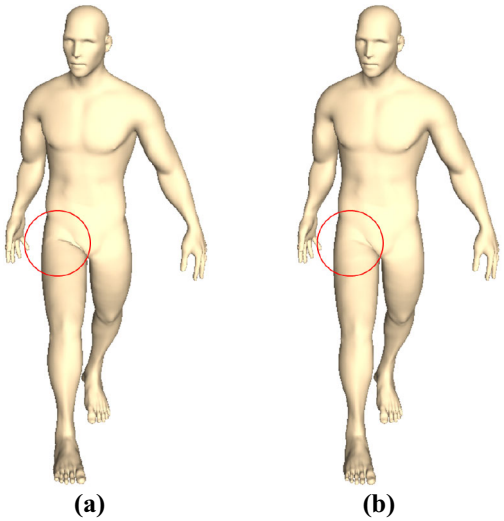


Fig. 16 Comparison of our method with the latest PBD-based skinning approach. **a** Method in [31], **b** our method

In the first experiment, we compare our skinning technique with the linear blend skinning (LBS) and dual quaternion skinning (DQS) approaches in joint bending and twisting (Fig. 15). Since LBS suffers from the volume collapse and DQS suffers from the bulging artefacts, the visual performance of our skinning method is the best.

We also compare our method with the latest PBD-based skinning approach [31]. It is the deformation of a human character at the same pose. The result is illustrated in Fig. 16. The data size and the computation time are compared in Table 1. Since the self-collision constraint is considered in our method, it can solve the intersection problem in the contact regions (groin) compared with method in [31].

Table 1 Data size and time cost (ms) of the method in [31] and ours

Method	Number of vertices	Number of tetrahedra	Constraints	Computation time (ms)
Rumman [31]	24,461	51,110	157,960	18.3
Ours	24,461	51,110	233,531	22.4

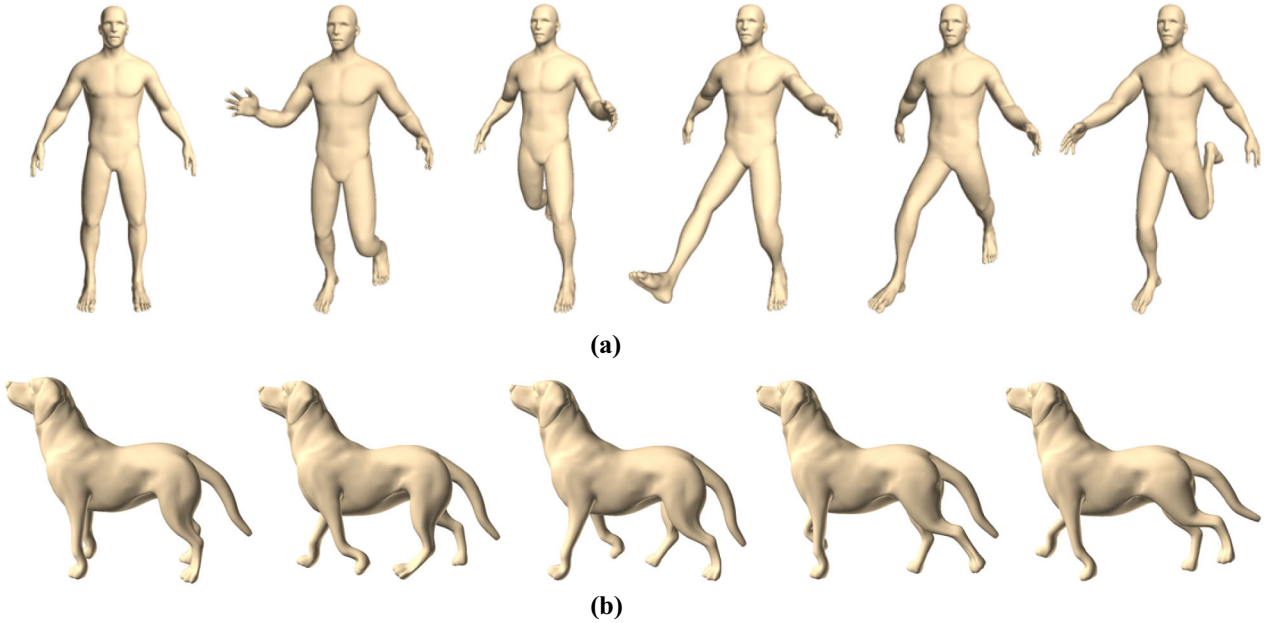


Fig. 17 Our method automatically produces the realistic skin deformations of the articulated characters. **a** Human model (184k constraints, 17.74 fps), **b** dog model (109k constraints, 25.6 fps)

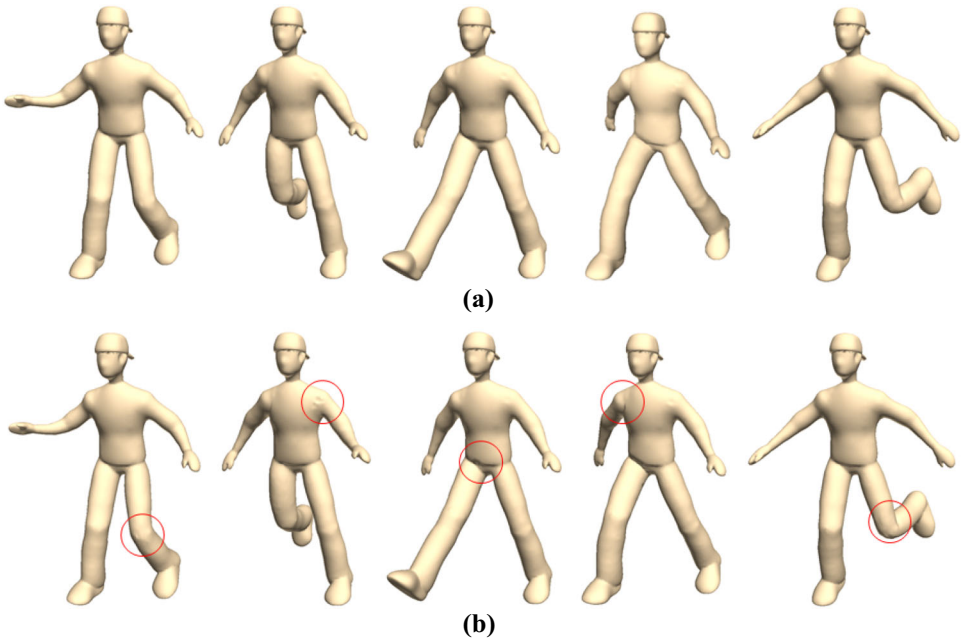


Fig. 18 Key frames of deformed doll model at the same poses. **a** With the computed and modified weights, **b** with the retargeted weights

In the second experiment, we demonstrate our automatic skinning method to animate two articulated characters (human and dog). Figure 17 illustrates the result. Due to the geometric and physical constraints in extended PBD, the deformation of joints area is smooth and realistic. To test our weight retargeting algorithm, we also transfer the painted weights from a human model (Fig. 2) to both a low mesh resolution model (doll) and a high mesh resolution model

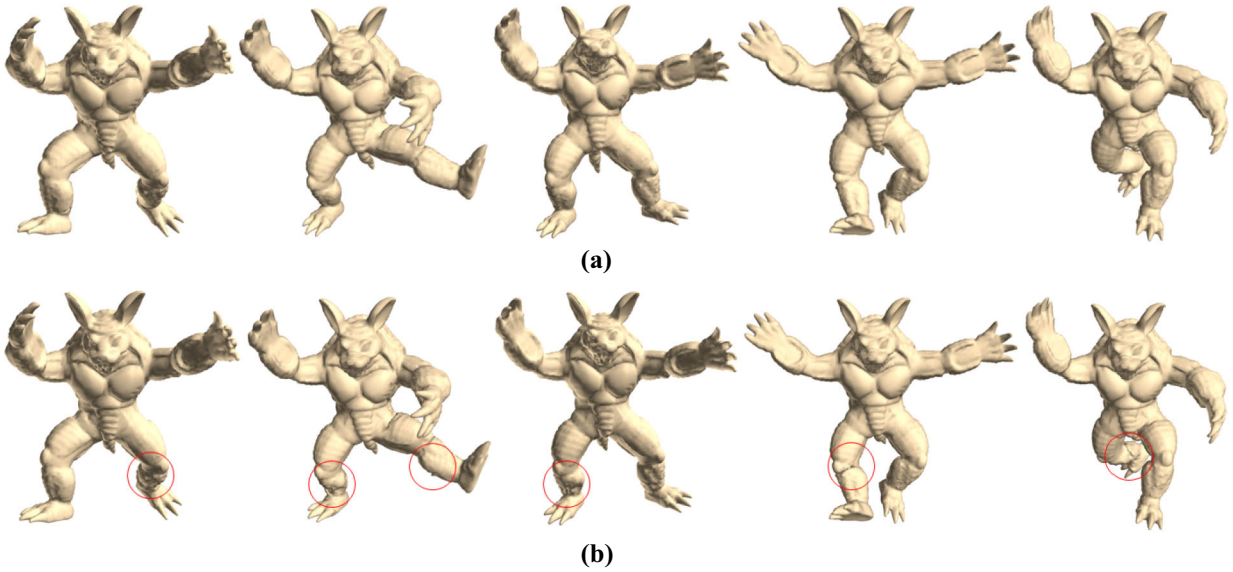


Fig. 19 Key frames of deformed armadillo model at the same poses. **a** With the computed and modified weights, **b** with the retargeted weights

Table 2 Data size and time cost (millisecond) of LBS, DQS and our method

Model	Number of vertices	Number of tetrahedra	LBS (ms)	DQS (ms)	Ours (ms)
Human	24,461	51,110	5.0	15.6	22.4
Dog	18,114	30,790	3.4	11.7	19.0
Doll	13,336	25,768	2.3	10.5	16.5
Armadillo	15,116	23,916	2.8	11.5	17.7

(armadillo), respectively. With the retargeted weights, the model is deformed by our automatic skinning method and compared with the animation using weights directly computed by Eq. (2) and modified by the animator. Figures 18 and 19 illustrates the result. From these figures, though there are small differences in several joints regions (red circles) between the animation using computed and manually refined weights, the retargeted weights still perform realistic skin deformation. Table 2 documents the data size and the computation time to animate four articulated characters by our automatic skinning method. Though there is an extra time cost for the iteration of extended PBD, due to the acceleration by parallel computation, the computation speed of our skinning approach maintains real time.

7 Conclusion and future work

In this paper, we have presented an automatic skinning and weight retargeting approach for articulated characters. The skin is deformed by the linear blend skinning method with an embedded skeleton at first. Then, we improve the traditional position-based dynamics (PBD) method by adding energy constraints and employ a number of geometric and physical constraints to refine the deformed skin in joint areas automatically. This extended PBD-driven skinning refinement can reduce animators a significant amount of time from traditional rigging processes. Finally, to further liberate animators from the burden of tedious weight painting, we have developed the weight retargeting method using the surface matching and interpolation technique based on the biharmonic distance. With our new method, we could transfer the well-painted weight from an existing character model to a new one with the same topology. This process would ensure that the high-quality rig can be reused without any manual intervention. Through the parallel computation acceleration, our skinning approach maintains real time.

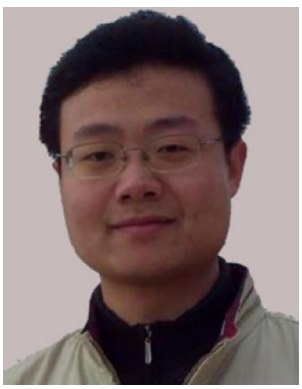
Nevertheless, our method is not without limitations. Though our extended PBD-based skinning method can fix the artifacts in joints area automatically, it needs a basically correct weight setting beforehand. Otherwise, it may generate non-smooth skin surface occasionally when large degree deformation happens (Fig. 11). To perform a perfect rig for the character, the accurate weight fixing by animators hands is still necessary occasionally. That is why we presented the weight retargeting technique to release the animator from

tedious weight fixing process. For weight retargeting, our method requires the retargeting model has the same skeleton topology with the original model. In future, we plan to extend our technique to the models with different skeleton topology.

Acknowledgements This research is supported by National Natural Science Foundation of China (No. 61402025, 61672149, 61532002).

References

1. Hiebert, B., Dave, J., Kim, TY., et al.: The Chronicles of Narnia: the lion, the crowds and rhythm and hues author video presentations are available from the citation page. In: ACM SIGGRAPH 2006 Courses, ACM, New York (2006)
2. Smith, J., White, J.: BlockParty: modular rigging encoded in a geometric volume. Acm Trans. Gr. **25**, 115 (2006)
3. Pan, J., Yang, X., Xie, X., et al.: Automatic rigging for animation characters with 3D silhouette. Comput. Anim. Virtual Worlds **20**(2–3), 121–131 (2009)
4. Baran, I., Popović, J.: Automatic rigging and animation of 3D characters. Acm Trans. Gr. **26**(3), 72 (2007)
5. Jacobson, A., Baran, I., Popović, J., et al.: Bounded biharmonic weights for real-time deformation. Acm Trans. Gr. **30**(4), 76–79 (2011)
6. Dionne, O., Lasa, MD.: Geodesic voxel binding for production character meshes. In: Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation ACM, New York, 173–180 (2013)
7. Magnenat-Thalmann, N., Laperrière, R., et al.: Joint-dependent local deformations for hand animation and object grasping. Proc. Gr. Interface 26–33, (1988)
8. Forstmann, S., Ohya, J.: Fast skeletal animation by skinned arc-spline based deformation. In: Fellner, D., Hansen, C. (eds.) Proceedings in EG Short Papers, The Eurographics Association. doi:10.2312/egs.20061014 (2006)
9. Mukai, T., Kuriyama, S.: Efficient dynamic skinning with low-rank helper bone controllers. Acm Trans. Gr. 35(4), (2016)
10. Kavan, L., Collins, S., Zra, J., et al.: Geometric skinning with approximate dual quaternion blending. Acm Trans. Gr. **27**(4), 995–999 (2008)
11. Kim, Y.B., Han, J.H.: Bulging-free dual quaternion skinning. Comput. anim. virtual worlds **25**(3–4), 321–329 (2014)
12. Vaillant, R., Barthe, L., Guennebaud, G., et al.: Implicit skinning: real-time skin deformation with contact modeling. Acm Trans. Gr. **32**(4), 96–96 (2013)
13. Vaillant, R., Guennebaud, G., Barthe, L., et al.: Robust iso-surface tracking for interactive character skinning[J]. Acm Trans. Gr. **33**(6), 189 (2014)
14. Müller, M., Chentanez, N.: Adding physics to animated characters with oriented particles, the workshop on virtual reality interactions and physical simulations. Vriphys 2011, Lyon, France, DBLP 83-91, (2011)
15. James, D.L., Twigg, C.D.: Skinning mesh animations. Acm Trans. Gr. **24**(3), 399–407 (2005)
16. Park, S.I., Hodgins, J.K.: Data-driven modeling of skin and muscle deformation. Acm Trans. Gr. **27**(3), 15–19 (2008)
17. Lewis, J.P., Cordner, M., Fong, N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. Conf. Comput. Gr. Interact. Tech. 165–172, (2000)
18. Xu, H., Barbič, J.: Pose-space subspace dynamics. Acm Trans. Gr. **35**(4), 1–14 (2016)
19. Mohr, A., Gleicher, M.: Building efficient, accurate character skins from examples. Acm Trans. Gr. **21**(3), 562–568 (2003)
20. Merry, B., Marais, P., Gain, J.: Animation space: a truly linear framework for character animation. Acm Trans. Gr. **25**(4), 1400–1423 (2006)
21. Seo, J., Seol, Y., Wi, D., et al.: Rigging transfer. Comput. Anim. Virtual Worlds **21**(3–4), 375–386 (2010)
22. Ju, T., Zhou, Q.Y., Panne, M., et al.: Reusable skinning templates using cage-based deformations. Acm Trans. Gr. **27**(5), 32–39 (2008)
23. Le, B.H., Hodgins, J.K.: Real-time skeletal skinning with optimized centers of rotation. Acm Trans. Gr. **35**(4), 1–10 (2016)
24. Allen, B., Curless, B., Popović, J., et al.: The space of human body shapes: reconstruction and parameterization from range scans[J]. Acm Trans. Gr. **22**(3), 587–594 (2003)
25. Ali-Hamadi, D., Liu, T., Gilles, B., et al.: Anatomy transfer. Acm Trans. Gr. **32**(6), 188 (2013)
26. Avril, Q., Ribet, S., Ghafourzadeh, D. et al.:Animation setup transfer for 3D characters. Comput. Gr. Forum 115–126, (2016)
27. Müller, M., Heidelberger, B., Hennix, M., et al.: Position based dynamics. J. Vis. Commun. Image Representation **18**(2), 109–118 (2007)
28. Bender, J., Müller, M., Otaduy, M.A., et al.: A survey on position-based simulation methods in computer graphics. Comput. Gr. Forum **33**(6), 228–251 (2014)
29. Pan, J., Bai, J., Zhao, X., et al.: Real-time haptic manipulation and cutting of hybrid soft tissue models by extended position-based dynamics. Comput. Anim. Virtual Worlds **26**(3–4), 321–335 (2015)
30. Macklin, M., Ller, M., Chentanez, N., et al.: Unified particle physics for real-time applications. Acm Trans. Gr. **33**(4), 1–12 (2014)
31. Rumman, N.A., Fratarcangeli, M.: Position-based skinning for soft articulated characters. Comput. Gr. forum **34**(6), 240–250 (2015)
32. PhysX-Nvidia. http://physxinfo.com/wiki/
33. Lipman, Y., Rustamov, R.M., Funkhouser, T.A.: Biharmonic distance. Acm Trans. Gr. **29**(3), 483–496 (2010)
34. Yen, L., Fouss, F., Decaestecker, C., et al.: Graph nodes clustering based on the commute-time kernel, in proceedings of the 11th Pacific-Asia conference on knowledge discovery and data mining (PAKDD 2007). Lect. notes in Comput. Sci. **4426**, 1037–1045 (2007)
35. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. ACM Natl. Conf. **23**, 517–524 (1968)



Junjun Pan received both BSc and MSc degree in School of Computer Science, Northwestern Polytechnical University, China. In 2006, he studied in National Centre for Computer Animation (NCCA), Bournemouth University, UK as PhD candidate with full scholarship. In 2010, he received PhD degree and worked in NCCA as Postdoctoral Research Fellow. From 2012 to 2013, he worked as a Research Associate in Center for Modeling, Simulation and Imaging in Medicine, Rensselaer Polytechnic Institute, USA. In Nov 2013, he was appointed as Associate Professor in School of Computer Science, Beihang University, China. His research interests include virtual surgery and computer animation.

Lijuan Chen is a master student in Beihang University. From 2014, she studied in the State Key Laboratory of Virtual Reality Technology and Systems in China. Her research interests include computer animation and virtual reality.

Yuhan Yang is a master student in Beihang University. From 2016, he studied in the State Key Laboratory of Virtual Reality Technology and System in China. His research interests include physical simulation and virtual reality.

Hong Qin is a full professor of Computer Science in the Department of Computer Science at Stony Brook University (SUNY). He received his BS and his MS in Computer Science from Peking University, China. He received his PhD in Computer Science from the University of Toronto. Currently, he serves as an associate editor for the Visual Computer, Graphical Models, and Journal of Computer Science and Technology. His research interests include geometric and solid modeling, graphics, physics-based modeling and simulation, computer-aided geometric design, human-computer interaction, visualization, and scientific computing.