Skeleton driven animation based on implicit skinning

Jituo Li*, Guodong Lu

Institute of Engineering and Computer Graphics, Department of Mechanical Engineering, Zhejiang University, Hangzhou 310027, PR China

Abstract

Skeleton-driven animation methods have been commonly used in animating 3D characters. In these methods, a skinning process that binds the character surface onto the skeleton is required. This process is usually accomplished manually and is a time-consuming task. In this paper, we propose a novel method for automatically skinning skeletal character models. Given the motion of a skeleton, our method can animate the character model automatically. In our method, each joint coordinate is parameterized by its surrounding local surface. In such a way, the character's surface is implicitly bound onto the skeleton. Character animation is achieved by minimizing an energy function that is carefully designed to prevent unnatural volume changes and to guarantee smooth deformations. Experiments demonstrate the efficiency and excellent performance of our method.

1. Introduction

Because they are intuitive and efficient, skeleton-driven animation methods are often used for animating 3D characters. In these methods, the motion on each surface vertex is computed as a weighted combination of the motions on the bones. Setting the influencing weights of each bone on the surface vertices is a core technique of these methods, which are also known as skinning, vertex weighting or skeleton subspace deformation [1,2]. In most commercial animation packages [3], skinning is left to the user to be accomplished manually and is usually a time-consuming task. Moreover, using linear skinning to simulate nonlinear skin deformation makes the results prone to cause undesirable deformations around the joints, such as the deformations known as “collapsing elbow” and “candy wrapper”.

In this paper, we present a new skeleton-driven character animation approach. Our approach can achieve the skinning process automatically and reduces deformation artifacts effectively. Our approach has the following highlights:

1. Rather than skinning the character's surface explicitly by representing the vertex motions with bone motions in traditional methods, in our approach, joint positions are parameterized by surface vertex positions so that a character's surface is implicitly bound onto its skeleton. This implicit skinning can be achieved automatically.

2. Rather than deforming each vertex independently, as in traditional methods, our approach deforms the character surface by minimizing a Laplacian mesh deformation energy with volumetric constraints. The character's surface can be deformed smoothly, and undesirable volume changes around the joints are prevented.

The remainder of this paper is structured as follows. After reviewing related studies in Section 2, our implicit skinning method is proposed in Section 3. A skeleton-driven Laplacian mesh deformation method is proposed in Section 4. Experimental results and analysis are given in Section 5. The paper is concluded in Section 6.

2. Related work

A core research problem in skeleton-driven animation entails how to alleviate or avoid the manual labor of skinning. To address this issue, a variety of methods have been proposed. Example-based methods [4–8] determine how bones provide influencing weights from a set of examples of deformed character models. Heat diffusion based techniques [9] compute the influencing weights of bones by solving a heat equilibrium equation. Linear skinning methods and their variations use linear techniques to express the intrinsic nonlinear relationship between the character surface and its underlying skeleton; undesirable artifacts unavoidably appear in some cases. Additional methods that have been proposed include curved-skeleton-based methods [10,11], sweep-based methods [12], methods that add additional joints [5], spherical blend skinning methods [13], quaternion blending methods [14], and moving-least-squares-based methods [15]. The basic problem of each of these methods is that each vertex is updated independently while ignoring the local surface attributes such as connectivity constraints among adjacent vertices.

* Corresponding author.
E-mail addresses: jituo_li@zju.edu.cn, jituo-li@163.com, zju-ljituo@163.com (J. Li).
By preserving the local surface differential attributes, gradient domain based mesh deformations such as Laplacian mesh deformation methods [16–18] or the Poisson mesh deformation methods [19] can produce smooth deformation results with good local surface detail preservation. General gradient domain methods are surface based. They have a common problem that in large scale deformation serious volume change might occur, especially in regions around the joints. For addressing this issue, volumetric constraints are necessary. By converting a surface mesh into a tetrahedral mesh, Zhou et al. [20] deformed a character surface using a volumetric graph-based Laplacian deformation. Huang et al. [21] further developed this technique so that the model volume could be precisely preserved during model deformation.

Barycentric based methods have been proposed for either 2D space deformation [22] or 3D space deformation [23] using either mean-value coordinates [24,25], harmonic coordinates [26,27], or Green coordinates [28]. Barycentric based methods can parameterize a planar point with the vertices of a polygon enclosing the point or can parameterize a spatial point with the vertices on a closed cage enclosing the point.

Zhang et al. [29] combined skeleton driven deformation with Laplacian mesh deformation. In their method, cross-sectional meshes created along the bone segment are severed as volumetric constraints. The character's surface is deformed by a Laplacian based method with the constraint that each cross-sectional mesh has the same transformation as its relative bone segment. Such constraints are reasonable for the cross-sectional meshes in the middle of the bone segments; however, for the cross-sectional meshes near the joints, such constraints conflict with the fact that character surface deformation around joints is influenced by the bones linking the joint, collectively. As a result, undesirable deformation is likely to occur around the joint in character deformation.

In summary, gradient domain based methods and barycentric based methods are superior to traditional skeleton-driven methods in model deformation. However, these methods are less intuitive than the skeleton-driven methods. Combining the skeleton-based deformation and the gradient domain based deformation (or the barycentric based deformation) can make the best use of both the methods. Our approach practises this strategy. Rather than deforming a character's surface by rigidly maintaining the shapes of the cross-sectional meshes [29], our approach uses radial edges created on a skeleton to preserve the radial distance from the character surface to the skeleton. The rigidity of the radial edges is tunable, which makes the character deformation move smoothly along the skeleton while avoiding undesirable volume changes. In our approach, the skinning process and the radial edge creation can be achieved automatically. Thus, given the motion of the skeleton, the character model can be animated automatically.

3. Implicit skinning

A character skeleton is a composition of a series of bones that chain at the joints. We assume that the character skeleton is given. If not, the skeleton can be either created automatically [30–32] or created manually through commercial software [3]. Because the shape of the skeleton is determined by its joint locations, when we set up the correlation between the character surface and the joints, we can bind the surface onto the skeleton. In our approach, each joint's position is parameterized by the positions of an appropriate set of surface vertices around it. Anatomically, these vertices have transformations that are similar to the joints' transformations during deformation; these vertices are named joint-influencing-vertices.

Because the transformation of a joint is determined by the bones linking the joint, each joint's joint-influencing-vertices are selected from the boundary of the influencing regions of the bones that link at the joint. The bone-influencing-region is the region around a bone for which its transformation is mostly determined by the bone during animation. We use a skeleton-based mesh segmentation method [33] to decompose a character's surface into several regions. Each region remains a bone-influencing-region, as shown in Fig. 1(b), where each region is colored differently. The joint-influencing-vertices of a joint are selected from the vertices around the joint using one of the following three methods, depending on the number of bones on the joint.

For a joint linking two bones, a plane passing through the joint is used to fit the boundary between its two bone-influencing-regions. Their intersection is a cross-sectional polygon that has vertices taken from the intersection points. Such a polygon is named a joint-cross-sectional-polygon. The gray rectangles shown in Fig. 1(c) are fitting planes. Fig. 1(d) shows the joint-cross-sectional-polygons, in the figure, where line segments linking the polygon vertices and the joint are used to reflect the distribution of the polygon vertices. The joint position can be represented with polygon vertices using the mean value technique [22]. However, the polygon might be concave; in such a case, the mean value

Fig. 1. Decomposing a character model into several bone-influencing-regions: (a) an input model, (b) decomposing result, (c) planes fitting the boundaries between bone influencing regions, and (d) joint-cross-sectional-polygons. Line segments linking the polygon vertices and the relevant joints are drawn. (e) For each joint-cross-sectional-polygon, a minimal convex hull is created and is further refined and (f) the new convex hulls are used to parameterize the joint coordinates. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)
coordinates can contain negative weights, which have the potential to cause undesirable deformations. For addressing this issue, we extract a set of vertices from the polygon to form a new convex polygon with which to parameterize the joint. The new convex polygon is created as follows. The minimal convex hull of the cross-sectional polygon is extracted, as shown by the polygon within the green box in Fig. 1(e). Then, the shape of the minimal convex hull is refined by removing the short edges, as shown by the polygon within the blue box in Fig. 1(e). Suppose that \( \{p_k\} \) are the vertices on the new convex polygon for joint \( J_i \); then, \( J_i \) can be parameterized, as follows:

\[
J_i = \sum_k \omega_k p_k, \quad \sum_k \omega_k = 1
\]

where the bold letters denote the point positions, e.g., \( p_k \) for \( p_k \) and \( J_i \) for \( J_i \). In addition, \( \{\omega_k\} \) is the mean value coordinate of \( J_i \) in the new convex polygon and is computed using the method in the literature [22]. As \( p_k \) is located on a certain edge of the mesh, it can be linearly interpolated by the two end vertices of the edge. Therefore, \( J_i \) can be represented as a linear combination of mesh vertex positions, as follows:

\[
J_i = \sum_k \lambda_k v_k
\]

where \( \lambda_k \) is a non-negative coefficient, and \( v_k \) is the coordinate of vertex \( v_k \) on the mesh surface. The mesh vertices on the intersecting edges form the joint-influencing-vertices for joint \( J_i \).

For the case where a joint links only one bone, the surrounding mesh vertices that fall into an influencing sphere are selected as joint-influencing vertices. The center and the radius of the influencing sphere are set to be the joint position and twice the mesh's average edge length, respectively. The sphere radius can be adjusted interactively, if needed.

For the case where a joint links more than two bones, an influencing cylinder is set up with a central axis that passes through the joint; the radius of the cylinder is twice the mesh's average edge length. The joint-influencing-vertices are selected as the mesh vertices inside the cylinder. The central axis of the influencing cylinder is initially set to be parallel with the norm of the plane that fits the linking bones on the joint. The cylinder radius and the central axis orientation can be adjusted interactively, if needed.

Fig. 2(a) shows the joint-influencing-vertices for the last two cases. For these two cases, a 3D minimal convex hull for the joint-influencing-vertices can be created using the quick hull algorithm [34], as shown in Fig. 2(b). The joint positions are parameterized by the 3D convex hulls with a mean value technique [24]. Because the vertices on the convex hulls are from the mesh vertices, the joint coordinates can be represented as linear combinations of mesh vertices, as in Eq. (2).

With Eq. (2), the mesh surface is correlated with the joints. In character animation, \( J_i \) can be parsed from the skeleton motion data in each frame, and \( v_k \) is to be determined. The following joint location energy, \( E_L \), is expected to be as small as possible in each frame.

\[
E_L = \sum_{i=0}^{N_j} \left( \sum_k \lambda_k v_k - J_i \right)^2
\]

where \( N_j \) is the number of joints.

With Eqs. (2) and (3), the correlation between the character surface and its underlying skeleton is established; in other words, the skinning process is achieved automatically. It is an implicit skinning, because the vertex positions are implicitly represented by the joint positions.

4. Mesh deformation

4.1. Laplacian mesh deformation

In Eqs. (2) and (3), each joint is only correlated to a small number of mesh vertices. We use the Laplacian mesh deformation to extend its influence throughout the whole mesh surface. The Laplacian of a mesh computes the difference between each vertex \( v_i \) and its neighboring vertices [16,17]:

\[
\delta_i = L(v_i) = v_i - \sum_{j \in N(i)} w_{ij} v_j, \quad \sum_{j \in N(i)} w_{ij} = 1
\]

where \( L(v_i) \) and \( \delta_i \) are the Laplacian operator and Laplacian coordinate of \( v_i \) respectively, and \( N(i) = \{ j \mid (i,j) \in E \} \) is the number of adjacent edges on vertex \( v_i \). Weights \( w_{ij} \) can be chosen as cotangent functions [35,36]:

\[
w_{ij} \propto \cot \alpha_i \cot \beta_i
\]

where \( \alpha_i \) and \( \beta_i \) are the two angles opposite to the edge \( v_i v_j \) in the two triangles sharing this edge.

When the mesh is deformed, its Laplacian deformation energy \( E_L \) could be represented as

\[
E_L = \sum_{i=0}^{N_v} \left( \delta_i - \delta'_i \right)^2
\]

where \( N_v \) is the number of vertices on the mesh, \( v'_i \) is the deformed \( v_i \), \( \delta'_i \) is the Laplacian coordinate of \( v'_i \) and can be computed as

\[
\delta'_i = \mathbf{T}_i^T \delta_i
\]

where \( \mathbf{T}_i^T \) is a transformation matrix that transforms \( v_i \) to \( v'_i \). In our approach, \( \mathbf{T}_i^T \) is computed from the skeleton motion in each frame, which will be detailed in Section 4.3.

By combining Eqs. (3) and (5), a character model can be deformed by

\[
\min(E_E,E_L) = \mu_1 E + \mu_2 E_L
\]

where \( \mu_1 \) and \( \mu_2 \) are the two weights used to balance the importance of the two energy terms.

Fig. 3 shows an experimental result using Eq. (7) with \( \mu_1 = 1.0 \) and \( \mu_2 = 2.0 \). In this figure, the right arm surface deviates severely from the skeleton, and the model volume around the left elbow is changed undesirably. Such an undesirable result indicates that it
is insufficient to control the character surface by only constraining the joint positions. To avoid undesirable deformations, the key concept is to preserve the spatial relationship between the character surface and the skeleton in the deformation.

4.2. Volumetric constraints

Radial edges, line segments that link the character surface and the skeleton, are created to preserve the radial distance between the character surface and the skeleton. There are two types of radial edges given as follows:

1. **Radial edges on joints:**
   For each joint-cross-sectional-polygon and its relevant joint $J_i$, two orthogonal lines passing through $J_i$ are used to intersect the polygon to get four intersection points $(q_i^k), k=0,1,2,3$. The variables $(q_i^k)$ and $(q_i^k), k=0,1,2,3$ are called joint-radial-points on $J_i$.

2. **Radial edges on bones:**
   On each bone segment, we equidistantly sample several points called skeletal points, $(S_i)$. From each $S_i$, two orthogonal lines that are perpendicular to the bone segment are created. These lines intersect with the bone-influencing-region, which results in at most four intersection points, $(q_i^k), k=0,1,2,3$ and consequently at most four radial edges, as shown in Fig. 4(b). These intersection points are called bone-radial-points.

By attempting to maintain the length of each radial edge, the spatial relationship between the character surface and the skeleton can be preserved. Thus, the following two positional difference energies, $E_{pos1}$ on joints and $E_{pos2}$ on bones, are expected to be as small as possible in the deformation.

$$E_{pos1} = \sum \sum |2 \sum q_{m_j} - (J_i + T_i^k(q_i^k - J_i))|^2$$  \hspace{1cm} (8)

$$E_{pos2} = \sum \sum |2 \sum q_{m_j} - (S_i + T_i^k(q_i^k - S_i))|^2$$  \hspace{1cm} (9)

In Eqs. (8) and (9), without loss of generality, we suppose that $q_i^k$ is located on triangle $f_m(v_m0, v_m1, v_m2)$, and $(\alpha_0, \alpha_1, \alpha_2)$ is the barycentric coordinate of $q_i^k$ in $f_m$. $T_i^k$ and $T_i^k$ are the transformation matrices on joint $J_i$ and the skeletal point $S_i$ in the deformation. The variables $v_{m_j}$, $J_i$, and $S_i$ are the positions of $v_{m_j}$, $J_i$, and $S_i$ after the deformation. Positions $(J_i + T_i^k(q_i^k - J_i))$ and $(S_i + T_i^k(q_i^k - S_i))$ are the target positions of the corresponding joint-radial-point and bone-radial-point. The weight $\psi_i$ in Eq. (9) is used to reflect the importance of constraining $q_i^k$ to superpose its target position in the deformation result.

For $\psi_i$ in Eq. (9), the larger the value of $\psi_i$ is, the more likely it is for the bone-radial-point to superpose its target position and the more rigid is the mesh surface around the skeletal point $S_i$ in the deformation. Therefore, by assigning different values to $\psi_i$, we can tune the rigidity of the mesh surface along the skeleton. It is reasonable to set the value of $\psi_i$ to entail decreasing values from the midpoint to the two joints on each bone so that character surface deformations around the middle of each bone are mainly determined by the bone while character surface deformations around each joint are influenced by the bones linking the joint together. Hence, $\psi_i$ is set as follows:

$$\psi_i = e^{-3.05 - r_i}$$  \hspace{1cm} (10)

where $r_i$ is the length ratio of $S_i$ on its relative bone.

4.3. Local transformation

Skeleton motion data, such as BVH files [37], can be used with our technique. Such data are usually created by motion capture equipment and are widely used within the computer animation society. From these data, the joint positions and the transformation matrix on each bone, i.e., $T_i^k$ in Eq. (9), can be parsed. The values of $T_i^k$ in Eq. (6) and $T_i^k$ in Eq. (8) are deduced from the bone transformations. For the computations, the local transformation on each bone is converted into a quaternion.

It is rational to set the local transformation of a joint as the spherical linear average of the transformations on its adjacent bones.

For a mesh vertex, its local transformation is computed by following the fact that the surface deformation around the middle of a bone is mainly determined by the bone motion, while the surface deformation around a joint is influenced by the motions of
bones linking the joint. As each mesh vertex \( v_i \) belongs to a certain bone-influencing-region, it relates to a certain bone. Without loss of generality, suppose that the relative bone for \( v_i \) is \( B_j \) and \( B_j \)'s two joints are \( J_{0,1} \). The local transformation on \( v_i \) is computed as follows. Project \( v_i \) onto the line segment \( J_{0,1} \) in the direction orthogonal to \( J_{0,1} \). Suppose that the mapping point is \( p_i \). Let \( r_i \) be the length rate of \( p_i \) on \( J_{0,1} \). Thus, \( r_i \) satisfies \( p_i = r_i J_{0,1} + (1 - r_i) J_{0,1} \). Then, the following is the case for the local transformation on \( v_i \):

1. It is set to be the transformation on \( B_j \) when one of the following conditions is met: \( 0.3 \leq r_i \leq 0.7 \); \( 0 \leq r_i \leq 0.3 \) and \( J_{0,1} \) links to only one bone; and \( 0.7 \leq r_i \leq 1.0 \) and \( J_{0,1} \) links to only one bone.
2. Otherwise, it is set to be a spherical linear interpolation between the transformations on the bones links joint \( J_{0,1} \) (if \( 0 \leq r_i \leq 0.3 \)) or \( J_{1} \) (if \( 0.7 \leq r_i \leq 1.0 \)) with the weight on the transformation of bone \( B_j \), which is \( \omega_j = (1 - (1/3))(r_i/0.3) + (1/n) \), where \( n \) is the number of linking bones on \( J_{0,1} \) and \( J_{1} \), and \( R_i = r_i \) if \( 0 \leq r_i \leq 0.3 \) or \( R_i = 1 - r_i \) if \( 0.7 \leq r_i \leq 1.0 \); the weights on the other bones are set to \( (1.0 - \omega_j)/n \).

With the above method, \( \delta_i \) in Eq. (6) can be updated efficiently.

### 4.4. Deformation computation

Combining all of the above energies, our approach deforms a character model by the following:

\[
\min E_1 = \mu_1 E_1 + \mu_2 E_2 + \mu_3 E_{\text{pos1}} + \mu_4 E_{\text{pos2}}
\]  

(11)

where the coefficients \( \mu_i \) (\( i = 1, 2, 3, 4 \)) are used to balance the importance of the four energy terms. The comparatively smaller \( \mu_1 \) is, the more likely it is to preserve the lengths of radial edges, hence to prevent the undesirable volume loss around the joints. However, the comparatively smaller \( \mu_4 \) is, the more likely it is to cause unsmooth deformation results. Experimental results are shown in Fig. 5 where a cylinder is bented with different values of \( \mu_4 \). After many experiments, we have found that it is appropriate to set \( \mu_1 = 1.0, \mu_2 = 2.0, \mu_3 = 2.0, \) and \( \mu_4 = 2.0 \). Such a set of coefficients can balance the volume preservation and surface smoothness in the deformation results well.

With Eq. (11), our approach is free to the collapsing elbow and candy-wraper, which are two well-known shortcomings of traditional skinning methods. Fig. 6 shows a comparison between the traditional skinning methods and our approach in bending and twisting a cylinder model.

In Eq. (11), \( E_{\text{pos1}} \) preserves the local surface details around the joints. Without this term, an undesirable volume change is prone to occur around the joint, as shown in Fig. 7(b) where a collapsing elbow is caused. \( E_{\text{pos2}} \) constrains the character surface around each bone, having a similar transformation with the bone. Without this term, the character surface would deviate from the skeleton undesirably, as shown in Fig. 3. As a comparison, the method proposed in the literature [29] is similar to setting \( \mu_2 \) and \( \mu_3 \) in Eq. (11) to zero and setting \( \psi_i \) in Eq. (9) to a constant, as a result, unnatural deformation occurs around the joints, as shown in Fig. 7(c) where the surface around the elbow looks too flexible as it should be in the real world. While by setting \( (\mu_1, \mu_2, \mu_3, \mu_4) \) in Eq. (11) to \( (1.0, 2.0, 2.0, 2.0) \), it yields good deformation results, as shown in Fig. 7(a) where the spatial relationship between the character’s surface and the skeleton is well preserved.

Eq. (11) is a quadratic function of the new vertex coordinates. This equation can be converted into a linear system by setting the gradient of \( E \) with respect to each new vertex coordinate to zero. Hence, we have

\[
A \mathbf{x} = \mathbf{b}
\]

(12)

where \( \mathbf{x} \) is the vector for \( m \) mesh surface vertices, \( \mathbf{b} \) is a known vector deduced from Eq. (11), and \( A \) is an \( n \)-by-\( m \) coefficient matrix. Also, \( n = m + s_1 + s_2 + s_3 \) where \( s_1, s_2, \) and \( s_3 \) are the number of joints, the number of joint-radial-points, and the number of bone-radial-points, respectively.

\( A \) is a sparse matrix, because each row of \( A \) is either deduced from a Laplacian operator on a certain vertex or is a mean value coordinate related to a small number of vertices. In addition, \( A^T A \) is sparse and positive definite. Eq. (12) can be solved by applying sparse Cholesky factorization to the associated normal equations, as follows:

\[
(A^T A) \mathbf{x} = A^T \mathbf{b}
\]

Fig. 6. Comparing our approach to the traditional skinning methods: a cylinder model in (a) is bented or twisted through 90° by our approach (b) and the traditional skinning methods (c). In (b) and (c), figures in the first row and the second row are the bending results and twisting results, respectively.
\((\mathbf{A}^T\mathbf{A})\) can be factored as
\[
(\mathbf{A}^T\mathbf{A}) = \mathbf{R}^T\mathbf{R}
\]  \hspace{1cm} (13)
where \(\mathbf{R}\) is an upper-triangular sparse matrix. Because \(\mathbf{A}\) is constant during animation, the factorization is computed only once. In each frame, \(\mathbf{x}\) can be solved efficiently by back substitution:
\[
\mathbf{R}^T\mathbf{\xi} = \mathbf{A}^T\mathbf{b} \text{ with } \mathbf{Rx} = \mathbf{\xi}
\]  \hspace{1cm} (14)

5. Experimental results and analysis

To evaluate the performance of our approach, more experimental results were obtained. In these experiments, skeleton motion data, in BVH files, are used to drive the models. Figs. 8 and 9 are used to indicate that our approach is competent for animating character models. In the second row of Fig. 9, different colors are used to indicate different bone-influencing regions. Fig. 10 is used to indicate that our approach can be used to animate multiply disjointed components, which provide a geometric method for clothing simulation. A video is also provided as a supplement. Fig. 11 is used to test the performance of our approach on character models with various mesh resolutions. As shown in the figures, our approach performs well both on high resolution models and on low resolution models.

Supplementary material related to this article can be found online at doi:10.1016/j.cag.2011.07.005.

As described in Section 4.3, the local transformation for each vertex \(\mathbf{T}_i^V\) in Eq. (6) is computed from the transformations on the bones. To evaluate the effectiveness of the \(\mathbf{T}_i^V\) computation, we use the As-Rigid-As-Possible (ARAP) [18] technique to optimize \(\mathbf{T}_i^V\). The ARAP technique optimizes the local transformation on the vertices with the aim that, for each local part of the model, it should be changed as rigidly and as smoothly as possible. The

![Fig. 7. Deformation results by setting \((\mu_1, \mu_2, \mu_3, \mu_4)\) in Eq. (11) to \((1.0, 2.0, 2.0, 2.0)\), \((1.0, 2.0, 0.0, 2.0)\), and \((1.0, 0, 0, 2.0)\). In (c), the coefficient \(\psi_i\) in \(E_{\text{mix}}\) in Eq. (11) is set to 1.](image)
ARAP technique builds on a nonlinear energy minimization. Fig. 12(a) and (b) shows the deformation results with our approach. We use the ARAP technique to optimize $T^j_i$ on the deformation results. Then, we use the optimized $T^j_i$ to deform the human model again; the new deformation results are shown in Fig. 12(c) and (d). From these figures, we find that the deformation results before and after the local transformation optimization are very close in shape. The deformation results with optimization could be a little bit smoother compared with the results without optimization, shown as the region near the right knee. However, such an optimization process might cause self-intersection around the joint when its joint angle is small, shown as the region near the right elbow in Fig. 12(d). Moreover, using the ARAP technique for optimization requires a singular value decomposition of a 3-by-3 matrix for each vertex, which is a computational burden for real-time animation. At the same time, our method needs only linear interpolation. These comparisons show that our local transformation computational method is effective and efficient.
In our experiments, when the pose of the skeleton changes extremely, self-intersection of the character model would happen, as shown in Fig. 13, if no additional treatment is made. Anatomically, when a joint bends, the skin around it will squeeze; therefore, this problem can be tackled by shortening the objective lengths of the radial edges in the squeezing direction and lengthening them in the orthogonal direction, so as to simulate the squeezing effect. The radial edges can either be adjusted interactively frame by frame by the user or can be adjusted automatically by the system when the self-intersections are detected. We would like to develop this technique in the future.

We have implemented our approach using C++ on a computer with an Intel(R) Core(TM)2 Quad CPU Q8300 @2.50 GHz and with 3.00 GRAM. We used the TAUCS library [38] to solve Eqs. (13) and (14). The factorization in Eq. (13) takes the bulk of the computation. This factorization is computed only once for each model. The back-substitution in Eq. (14) is usually very fast. Table 1 shows the computational time for each model used in this paper.

Though our method is efficient, it is not as efficient as traditional skinning methods. In traditional skinning methods, each vertex is deformed by a linear combination of rotation matrices; therefore the computational time for producing each animation frame is in direct proportion to the number of vertices, which makes the computation very efficient, e.g., for deforming the male model used in Fig. 11(a) and (b), it takes 51 and 15 ms, respectively, for each frame.

From the comparison between deformation effects, as shown in Fig. 6, we have found that, for the surface regions near the middle of Fig. 10. Animation result of a clothed female model.

Fig. 11. Deformations of a character model with various resolutions. The vertex numbers in (a), (b), (c), (d) are 75000, 23,739, 750, and 350, respectively.
the bones, the deformation results produced by our method and by the traditional skinning methods are close in shape. However, for the surface regions near the joints, the deformation results can be mostly different. In future work, we would like to propose an explicit–implicit hybrid skinning method. A character model would be decomposed into two groups of the sub-regions. The first group is for the sub-regions near to the midpoints of bones, and the second group is for the sub-regions near to the joints. We would use the traditional skinning methods to animate the first group, and use our implicit skinning method to animate the second group. For each group, the sub-regions could be deformed in a parallel manner. Thus, the whole process would be much more efficient than the current implicit method, especially for models with large numbers of vertices.

We would like to leave more control to the user. We would like to develop a tool for interactively tuning the coefficients, including $\mu$ in Eq. (10), and $\psi_i$ in (9), because a convincing animation result is ensured only if the coefficient values of the energy function are set appropriately.

Because current commercial animation software packages only support the explicit skinning method, it is worthwhile in the future to learn the bone influencing weights from the animation results produced by our implicit skinning. Then, the influencing weights can be used directly in the commercial software packages or can be edited interactively by an expert animator for reuse. In this way, the manual labor in the traditional skinning method can be greatly reduced.

6. Conclusions

In our approach, the character model surface is implicitly and automatically bound onto a skeleton by parameterizing each joint location with the local surface around the joint. The animation reality is guaranteed by minimizing an energy function in each frame. This energy function is deduced from joint locations, local surface details, and volumetric constraints, which can be minimized by solving a sparse linear equation system with a constant coefficient matrix. Our approach is novel and can produce plausible animation results for skeletal characters.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 60903145, 61070105), the Beijing Natural Science Foundation (No. 4102062) and the Research Fund for the Doctoral Program of Higher Education of China (No. 20100101110025). We would like to thank Nils Hasler for providing the male model.

References