Volume 34 (2015), Number 1 pp. 1–11 COMPUTER GRAPHICS *forum*

Position-based Skinning for Soft Articulated Characters

Nadine Abu Rumman[†1] and Marco Fratarcangeli [‡2]

[1]Sapienza University of Rome, Italy
[2] Chalmers University of Technology, Sweden

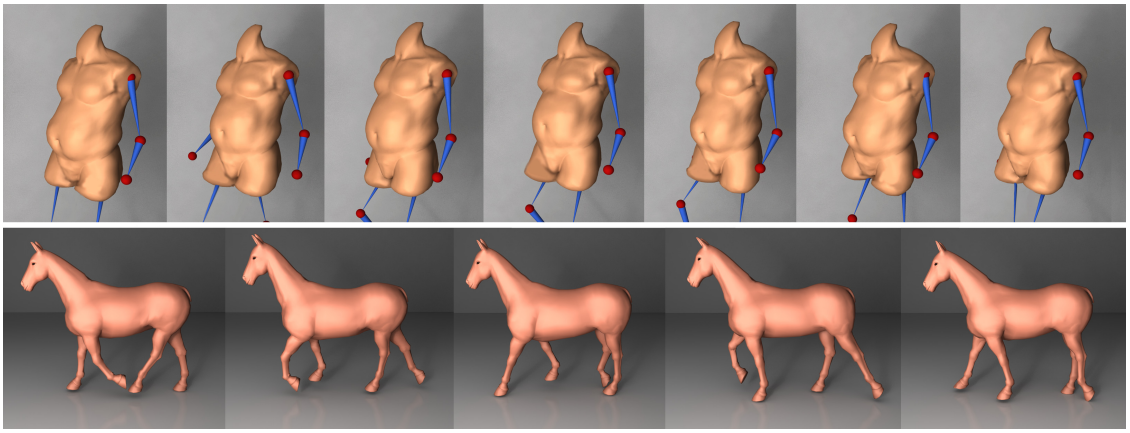


Figure 1: Real-time animations of complex articulated characters, including secondary motions and volume preservation. Top row: TORSO (11K constraints, 149 fps). Bottom row: HORSE (17K constraints, 130 fps).

Abstract

In this paper, we introduce a two-layered approach addressing the problem of creating believable mesh-based skin deformation. For each frame, the skin is first deformed with a classic linear blend skinning approach, which usually leads to unsightly artefacts like the well-known candy-wrapper *effect and volume loss. Then we enforce some geometric constraints which displace the positions of the vertices to mimic the behavior of the skin and achieve effects like volume preservation and jiggling. We allow the artist to control the amount of jiggling and the area of the skin affected by it. The geometric constraints are solved using a Position-Based Dynamics schema. We employ a graph coloring algorithm for parallelizing the computation of the constraints. Being based on Position-Based Dynamics guarantees efficiency and real-time performances while enduring robustness and unconditional stability. We demonstrate the visual quality and the performance of our approach with a variety of skeleton-driven soft body characters.*

Categories and Subject Descriptors (according to ACM CCS): 1.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation and Virtual Reality

1. Introduction

Animating articulated characters such as virtual humans and animals is still a central challenge in computer graphics and interactive applications [MZS*11, HTC*13, LYWG13]. Modeling compelling and believable skin deformations is difficult and computationally demanding due to the non-linear inner mechanics of the flesh. Physics-based skinning methods are able to reproduce realistic motions including secondary effects such as jiggling of soft tissues when the character is moving and volume conservation. Despite offer-

† e-mail:aburumman@dis.uniroma1.it
‡ e-mail:marcof@chalmers.se

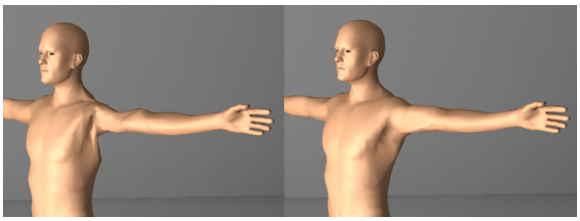


Figure 2: Two-layered deformer. *Left*. Step 1: Linear Blend Skinning (LBS) is applied for animating the rotation of the shoulder. Note the *candy-wrapper* effect. *Right*. Step 2: Vertex positions are adjusted using Position-Based Dynamics (PBD).

ing such realistic effects, physics-based simulation requires complex and intensive computations, and thus it is usually avoided in interactive animations such as video games. Furthermore, once the deformation parameters are specified, it is difficult to control the actual resulting shape of the character in every animation frame.

In this paper, we propose a simple and fast skinning method for soft character animation, suitable for real-time applications (Fig. 1). Our method is able to reproduce secondary effects and provides the artist with some level of control over them. We employ a two-layered deformation schema, the result of which approximates the behaviors of the skin. For each animation frame, the deformation of the character is decoupled in two steps. First, we apply classic Linear Blend Skinning (LBS, [MTLT88]) to the character. Then, in a second step, we solve a system of geometric constraints used to model in real-time the volume conservation and the jiggling of the skin. We simulate the skin as a soft body, and the deformation approach is based on Position-Based Dynamics schema (PBD, [MHHR07]). This approach is applied in a parallel fashion and solved on the multi-core CPU. The resulting skinned animations are unaffected by well known artefacts proper of LBS, namely the *candy-wrapper* effect and loss of volume (Fig. 2). Being based on PBD, this second step is efficient, controllable and unconditionally stable, even when large time steps are employed for advancing the character's dynamics.

Contributions In summary, we propose a practical method for animating mesh-based articulated characters in real-time. Our system does not explicitly model the mechanical properties of the human flesh as in typical physics-based systems; however it is still able to mimic the macro-behaviors of the skin, such as volume conservation and jiggling, while being relatively easy to implement and fast to compute. We also allow the artist to tune the amount of jiggling in specified areas of the skin.

2. Related Work

Our method is closely related to skeleton-based deformation techniques, soft character simulations and physically based skinning methods. We cover below the references in literature that are most relevant to our work.

Skeleton-based deformation techniques are classified into two categories; geometric methods and example-based methods. Geometric approaches to deforming articulated characters have shown reasonable results at interactive rates [MTLT88, SK00, FO06, VBG*13], starting from the standard real-time method "*skeletal subspace deformation*", also known as linear blend skinning (LBS) [MTLT88]. This method has been widely adopted in real-time applications such as games for its computational efficiency and straightforward GPU implementation [RLN06]. Unfortunately, linear blend skinning suffers from visual artefacts such as self-intersection, volume loss or the well-known "*candy-wrapper*" artefact, which are the result of the linear nature of the algorithm, since the linear interpolation of the transformation matrices is not equivalent to the linear interpolation of their rotations. An interesting extension of linear blend skinning called spline skinning comes from [FO06], which often produces better skinning deformations. Instead of using conventional matrix rotation, spline skinning represents each bone of the skeleton by a spline. The artefacts of LBS can be reduced by replacing the linear blending with a non-linear transformation blending (dual quaternion skinning) [KCZO07]. However, dual quaternion blending suffers from an undesirable joint-bulging artefact while bending, which requires artistic manual work to be fixed. Because fixing these artefacts manually is a tedious process, automatic skinning techniques [BP07, JBPS11] are becoming increasingly popular. In contrast, example-based methods remove these artefacts by adding pose examples [LCF00, KJP02, JT05, PH08] or additional skinning weights [WP02, MG03, MMG06]. An overview survey on linear skinning techniques can be seen in Jacka et al. [JRMG07]. Selecting good skinning weights is also critical to avoid deformation artefacts and to generate more natural deformations. Recently, an automatic computation of skinning weights was presented in [DdL13]. In their method, the influence weights are determined by using geodesic distances from each bone, making the inverse-distance weights shape-aware and can work with production meshes that may contain non-manifold geometry.

To preserve the volume and avoid self-intersections, von Funck et al. [vFTS06, AS07] presented approaches that deform the mesh vertices based on vector field integration. However, these approaches do not fit into the standard animation pipeline. Though cage-based skinning techniques are appealing techniques to achieve smooth deformations and preserve the volume of the skin geometry [LKCOL07, JZvdP*08], posing the cage requires manual manipulation of the cage vertices.

Recently, Kavan et al. [KS12, KH14] developed new skinning methods that avoid the artefacts of linear blend skinning as well as the bulging artefacts of dual quaternion skinning. All these techniques are either purely kinematic, lacking secondary motion effects like passive jiggling motion of fatty tissues, or example-based techniques that require the artists to create many different samples by hand for a wide variety of poses.

Soft character simulation The pioneering work of Terzopoulos et al. [TPBF87], encouraged the development of many physically based methods to simulate soft bodies or add dynamic effects to the skin [CHP89, TT93, LTW95, PSE03, MDM*02, CA005, Fra12]. A survey of Nealen et al. [NMK*06] provides comprehensive details of these techniques; however these methods are only valid for small deformations and are unsuitable for articulated characters' large deformations. A possible approach for physically based deformations of soft bodies is to focus on the surface rather than the volume. In particular Galoppo et al. [GOT*07] presented a fast method to compute the skin deformation on the surface of a soft body including a rigid core. Their formulation only considers the elastic energy from skin-layer deformation and does not include the deformation inside the volume, which may lead to inaccuracies when capturing pose-dependent deformations. In contrast, Capell et al. [CGC*02] used volumetric finite element mesh to represent the deformation of skin driven by the underlying skeleton motion. They extended their method to include rigging forces which guide the deformation to a desired shape [CBC*05]. In their method, they effectively handled the effect of skin movement by using skeletal constraints, but using forces that can violate the conservation of momentum may make their simulation unstable under large time steps. Recently, Kim and Pollard [KP11] proposed an approach relying on the finite element method (FEM) to simulate the skin deformation, able to handle both one-way and two-way simulations. Moreover, their method provides compelling secondary motions effects.

Physically based skinning Using physics in the skinning process enriches the purely geometric skeleton-based deformations with physically realistic secondary motions. The method of Shi et al. [SZT*08] is able to add realistic secondary deformation to the skeleton-based animations in real-time, by taking a surface mesh and a few sample sequences of its physical behavior. Kim and James [KJ11] proposed a domain-decomposition method to simulate articulated deformable characters entirely within a sub-space framework, where they combined locally-rotated non-linear subspace models to simulate the detailed deformations of the models. McAdams et al. [MZS*11] presented a robust method which provides convincing deformations of the skin using a uniform hexahedral lattice, where they introduce a one-point quadrature scheme and a multi-grid solver in order to improve the performance and stabilize the simulation. Deul and Bender [BMOT13] introduced a multi-layer character skinning based on shape matching with oriented particles, used to simulate the elastic behavior of a closed triangular mesh as representation of a skin model. They make a use of position-based constraints for coupling the skeleton with the skin and handling self-collisions.

Models of both [KP11] and [BMOT13] are more sophisticated and better reflect the inner structure of the character skin than ours. However, we decided to employ a robust combination of linear blend skinning and Position-based Dynamics, which requires less mathematical complexity and leads to believable animations with at least one order of magnitude faster computation time.

3. Position Based Skinning

The inputs of our method are a fine surface mesh representing the character in rest pose and an animated skeleton embedded within the mesh. From the surface mesh, we generate a tetrahedral mesh of the same shape using [BO05], which preserves the original outer surface geometry (Fig. 3).

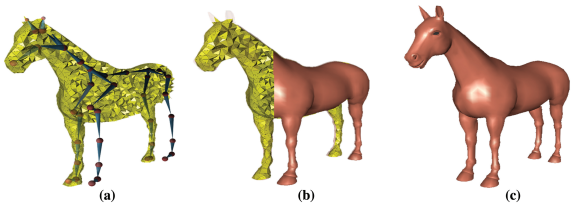


Figure 3: Inputs to our method; (a) the embedded skeleton within the tetrahedral mesh, (b) the fine surface corresponding to the tetrahedral mesh, and (c) the fine surface.

We use the tetrahedral elements for defining the geometric constraints used to compute the elastic motion of the character skin while moving, including volume preservation, in a way similar to [APH11]. In the initialization phase, the vertices of the original skin mesh are mapped to the tetrahedral elements by using barycentric coordinates, and the geometrical constraints are defined according to the rest pose of the character. Then, at every animation frame, the skeleton moves and the mesh vertices are first deformed through a standard linear blend skinning (LBS) process, and then their positions are adjusted by solving the constraints. The geometric constraints are solved using a Position-Based Dynamics schema. We employ a graph coloring algorithm for parallelizing the computation of the constraints. Finally, the resulting vertex positions are used for updating the fine mesh, which is employed for rendering. The whole mechanism is summarized in Fig. 4.

3.1. Linear Blend Skinning

We consider a skeleton as defined by its bones. Each bone is expressed as a rotation matrix. Given a bone j, we distinguish between its rest matrix $\mathbf{R}_j$ and its posed matrix $\mathbf{T}_j$

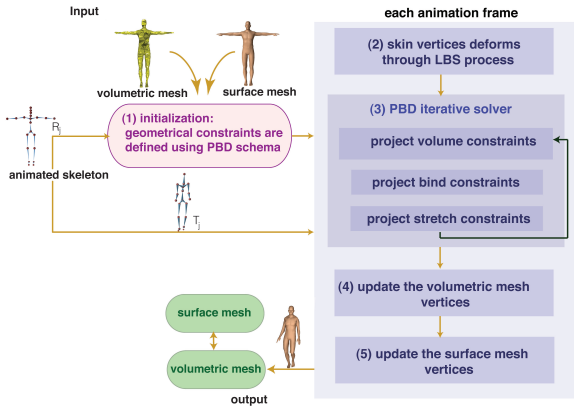


Figure 4: In the initialization phase, the fine surface mesh is converted to a tetrahedral mesh, which is used to define the soft geometric constraints. During the animation of the skeleton, the volumetric mesh is first deformed using Linear Blend Skinning, then the constraints are solved using a parallel Position-Based Dynamics schema.

(that is, its configuration in any animation frame), where $j = 1 \dots m$, and m is the total number of bones. For each animation frame, the motion of the posed bones is defined by a given input motion. At rest, the input mesh is associated with its skeleton. In LBS, each vertex $\mathbf{v}_i$ in the input mesh $\mathbf{v}_i$ is attached to one or more skeletal bones; each attachment affecting the vertex with a different strength, or *weight*. The final transformed vertex position $\mathbf{v}'_i$ is a weighted average of its initial position transformed by each of the attached bones, according to the following equation [MMG06]:

$$\mathbf{v}'_i = \sum_{j=1}^{m} w_{i,j} \mathbf{T}_j \mathbf{R}_j^{-1} \mathbf{v}_i \tag{1}$$

where the mesh vertices are $\mathbf{v}_i$ ... $\mathbf{v}_n$, and $\mathbf{v}_i$ is in homogeneous coordinates. $\mathbf{T}_j$ is the transformation matrix associated with bone j in its current pose, $\mathbf{R}_j^{-1}$ is the inverse transformation of the same bone in the rest pose, the bone transformations are represented using a list of matrices $\mathbf{T}_1 \cdots \mathbf{T}_m \in \mathbb{R}^{4\times4}$ and $w_{i,j}$ is the weight binding $\mathbf{v}_i$ to the bone j. The weights are normally assumed to be convex, so $w_{i,1} + \dots + w_{i,m} = 1$ and $w_{i,j} \geq 0$.

We attach each vertex $\mathbf{v}_i$ to at most 3 bones. To compute the weights, we use the following formula, which is based on the distance from the vertex to the bone:

$$w_{i,j} = \begin{cases} 1.0 & \text{if } \frac{1}{6} \leq \frac{(\mathbf{v}_i - \mathbf{s}_j)}{|\mathbf{d}_j - \mathbf{s}_j|} \cdot \frac{(\mathbf{d}_j - \mathbf{s}_j)}{|\mathbf{d}_j - \mathbf{s}_j|} \leq \frac{5}{6} \\ 1.0/n_{joints} & \text{otherwise} \end{cases} \tag{2}$$

where $\mathbf{s}_j$ and $\mathbf{d}_j$ are the parent and child joint positions (end points) of the bone j nearest to $\mathbf{v}_i$, and n_{joints} is the number of bones attached to the end points bone nearest to $\mathbf{v}_i$. In this way, vertices far away from the end points of a bone are rigidly influenced by only that bone, while vertices near the end points are equally influenced by the bones in that area. The resulting deformation is smooth but suffer from joint collapse issues which are fixed by the second step of our method, as described in the next section.

3.2. Position-based Dynamics

Position-Based Dynamics (PBD) [MHHR07, BMO*14] is a method based on Verlet integration for interactively animating deformable objects. The objects are modeled as a set of n particles whose motion is governed by a set of m non-linear constraints. The system of constraints is solved using Gauss-Seidel iterations by directly updating the particle positions. PBD avoids the use of internal forces, and the positions are updated such that the angular and the linear momenta are implicitly conserved. In this way, the whole process is not affected by the typical instabilities of interactive physically-based methods. To keep the paper as self contained as possible, in the following we first briefly summarize the core idea of PBD. Then, in Sections 3.2.1-3.2.3, we define the geometric constraints used in our model and how they are solved. Finally, in Section 3.3, we describe how the constraints are solved in parallel using a graph coloring algorithm.

The set of constraints is composed by non-linear equality and inequality equations such that:

$$C_i(\mathbf{p}) \succ 0, \quad i = 1, \dots, m \tag{3}$$

where the symbol $\succ$ stands for either $=$ or $\geq$, $\mathbf{p} = \left[\mathbf{p}_1^T, \dots, \mathbf{p}_n^T\right]^T$ is the vector of particle positions, n is the number of particles and m is the number of constraints. The constraints are generally non-linear and they are solved sequentially through Gauss-Seidel iterations. Each equation is linearized individually in the neighborhood of C around the current configuration $\mathbf{p}$ to find the correction $\Delta\mathbf{p}$:

$$C_i(\mathbf{p} + \Delta\mathbf{p}) \approx C_i(\mathbf{p}) + \nabla_{\mathbf{p}} C_i(\mathbf{p}) \cdot \Delta\mathbf{p} = 0 \tag{4}$$

where $\nabla_{\mathbf{p}} C_i(\mathbf{p})$ is the vector containing the derivatives of the equation C_i w.r.t. the n components of $\mathbf{p}$.

The correction $\Delta\mathbf{p}$ is imposed to be in the direction of $\nabla_{\mathbf{p}} C(\mathbf{p})$:

$$\Delta\mathbf{p} = \lambda_i \nabla_{\mathbf{p}} C_i(\mathbf{p}) \tag{5}$$

This condition implicitly conserves the linear and angular momenta, while at the same time allowing solving of the under-determined system of constraints. Combining Eqs. 4 and 5 yields:

$$\lambda_i = -\frac{C_i(\mathbf{p})}{|\nabla_{\mathbf{p}} C_i(\mathbf{p})|^2} \tag{6}$$

3.2.1. Stretch Constraint

We define one *stretch* constraint for the particles $(\mathbf{p}_1, \mathbf{p}_2)$ at the end points of each edge of the mesh, including the edges of the internal tetrahedrons (Fig.5):

$$C(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d = 0 \tag{7}$$

is used to keep particles $\mathbf{p}_1$ and $\mathbf{p}_2$ at distance d. where d is the rest length of the edge. Given the configuration $(\mathbf{p}_1, \mathbf{p}_2)$ of two particles connected by a stretch constraint, the corrections to the positions (Eq. 5) in order to satisfy the constraint are:

$$\begin{aligned} \Delta\mathbf{p}_1 &= -\tfrac{1}{2} k_s \left(|\mathbf{p}_1 - \mathbf{p}_2| - d\right) \tfrac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|} \\ \Delta\mathbf{p}_2 &= +\tfrac{1}{2} k_s \left(|\mathbf{p}_1 - \mathbf{p}_2| - d\right) \tfrac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|} \end{aligned} \tag{8}$$

where $k_s \in [0,1]$ is a stiffness scalar parameter which slows the convergence of the constraint and provides a "springy" behavior to the corresponding edge.

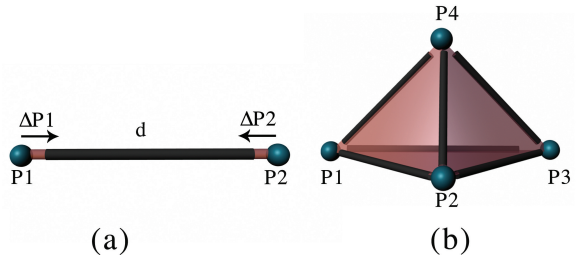


Figure 5: A volume constraint is defined for each tetrahedron, together with six stretch constraints (one for each edge).

3.2.2. Tetrahedral Volume Constraint

We define one *volume* constraint for the particles $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4)$ at the corners of each tetrahedral of the mesh. The volume constraint maintains the rest volume of four particles forming a tetrahedron, enforcing the conservation of the total volume of the character's body:

$$C(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) = \frac{1}{6}(\mathbf{p}_{2,1} \times \mathbf{p}_{3,1}) \cdot \mathbf{p}_{4,1} - V_0 \tag{9}$$

where $\mathbf{p}_{i,j}$ is the short notation for $\mathbf{p}_i - \mathbf{p}_j$ and V_0 is the rest volume of the tetrahedral.

The gradient with respect to each particle is:

$$\nabla_{\mathbf{p}_2} C(\mathbf{p}_{2,3}) = \frac{1}{6}(\mathbf{p}_{2,1} \times \mathbf{p}_{3,1}) \tag{10}$$

$$\nabla_{\mathbf{p}_3} C(\mathbf{p}_{3,4}) = \frac{1}{6}(\mathbf{p}_{3,1} \times \mathbf{p}_{4,1}) \tag{11}$$

$$\nabla_{\mathbf{p}_4} C(\mathbf{p}_{4,2}) = \frac{1}{6}(\mathbf{p}_{4,1} \times \mathbf{p}_{2,1}) \tag{12}$$

$$\begin{aligned} \nabla_{\mathbf{p}_1} C(\mathbf{p}_1) &= -\left(\nabla_{\mathbf{p}_2} C(\mathbf{p}_{2,3}) + \nabla_{\mathbf{p}_3} C(\mathbf{p}_{3,4}) + \right. \\ &\quad \left. + \nabla_{\mathbf{p}_4} C(\mathbf{p}_{4,2})\right) \\ &= -\frac{1}{6}(\mathbf{p}_{2,1} \times \mathbf{p}_{3,1} + \mathbf{p}_{3,1} \times \mathbf{p}_{4,1} + \\ &\quad + \mathbf{p}_{4,1} \times \mathbf{p}_{2,1}) \end{aligned} \tag{13}$$

The correction of each particle belonging to the tetrahedron is:

$$\Delta\mathbf{p}_1 = -\frac{1}{6} s \cdot k_v (\mathbf{p}_{2,1} \times \mathbf{p}_{3,1} + \mathbf{p}_{3,1} \times \mathbf{p}_{4,1} + \mathbf{p}_{4,1} \times \mathbf{p}_{2,1}) \tag{14}$$

$$\Delta\mathbf{p}_2 = \frac{1}{6} s \cdot k_v (\mathbf{p}_{2,1} \times \mathbf{p}_{3,1}) \tag{15}$$

$$\Delta\mathbf{p}_3 = \frac{1}{6} s \cdot k_v (\mathbf{p}_{3,1} \times \mathbf{p}_{4,1}) \tag{16}$$

$$\Delta\mathbf{p}_4 = \frac{1}{6} s \cdot k_v (\mathbf{p}_{4,1} \times \mathbf{p}_{2,1}) \tag{17}$$

where k_v is the stiffness parameter and s is the scaling factor:

$$s = \frac{\frac{1}{6}(\mathbf{p}_{2,1} \times \mathbf{p}_{3,1}) \cdot \mathbf{p}_{4,1} - V_0}{\Sigma_{i=1}^{4} \mid \nabla_{\mathbf{p}_i} C(\mathbf{p}_i) \mid^2} \tag{18}$$

3.2.3. Bind Constraint

We define a *bind* constraint between each particle and its nearest bone. Basically, a bind constraint is a stretch constraint between a particle and its projection on the nearest skeleton bone. When the skeleton moves and the joints rotate, the projection point for each particle is updated accordingly and the bind constraint pushes or pulls the particle to maintain the rest distance. This mechanism is depicted in Fig.6.

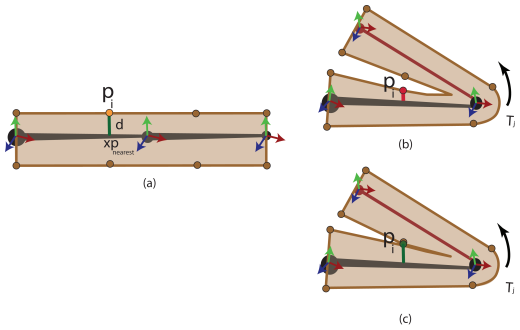
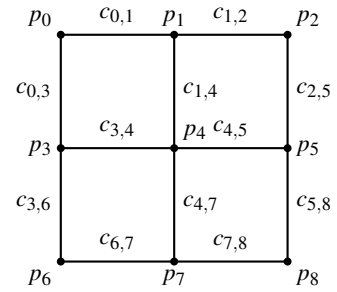


Figure 6: (a) The particle $\mathbf{p}_i$ in the rest pose, d is the rest distance from the nearest bone. (b) The particle $\mathbf{p}_i$ is displaced from the first step of our approach, by using LBS. (c) The *bind* constraint maintains the rest distance from the bone.

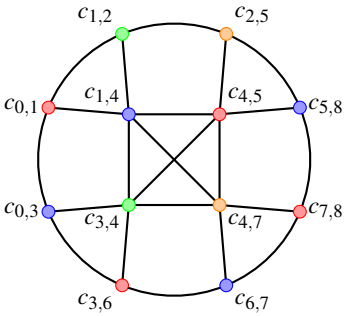
3.3. Parallel Position-Based Dynamics

The set of constraints in the original Position-Based approach are solved in a Gauss-Seidel fashion; each constraint is solved in a serial way, in succession, directly correcting the position of the particles. This process is iterated several times for each animation frame; for each iteration, the difference between the current and the optimal solution of the system decreases. This approach is efficient when the number of constraints is relatively small and thus the number of iterations needed for reaching a good approximation of the global solution is low. However, when a high number of constraints is involved, both the computational cost of a single iteration and the number of iterations for each frame increases. In this case, the serial solution of the set of constraints quickly becomes unsuitable for interactive animation.

As noted in [BMO*14], this process can be parallelized by using a graph coloring algorithm. A dual graph is defined, where each node represents a constraint and two nodes are connected if they share at least one particle. Then the graph is colored with a distance-1 algorithm, such that two adjacent nodes are not assigned with the same color. The constraints belonging to the same color do not share any particle and can be solved in parallel by using a number of dedicated CPU threads. When all the constraints belonging to a color are solved, the constraints assigned to the following color are solved, until all the colors are considered. The process of coloring the graph happens just once in the initialization phase, and it does not affect the performance during the animation. In this way the asymptotic complexity for each iteration improves from $\mathcal{O}(m)$ where m is the number of constraints, to $\mathcal{O}(q)$ where q is the total number of colors. We used Sequential Vertex Coloring [CM83] which usually leads to a number of colors equal to the size of the biggest clique in the graph, or slightly bigger. Fig. 7 depicts this mechanism for a simple example.



(a) A set of particles connected by stretch constraints. Here $c_{i,j}$ is the short notation for $C(\mathbf{p}_i, \mathbf{p}_j)$.



(b) The corresponding dual graph. Two constraints are connected if they share at least one particle. The graph is colored with a distance-1 algorithm and the set of constraints belonging to the same color are solved in parallel, reducing the number of serial steps from 12 to 4.

Figure 7: A graph coloring algorithm is applied to a simple particle system to parallelize the computation of the constraints; (a) a simple particle system of 9 particles and 12 constraints, (b) a dual graph is defined, where each node represents a constraint and two nodes are connected if they share at least one particle.

4. Soft Control

We allow the artist to specify the amount of jiggling in a manually specified area of the mesh surface. In a preprocessing step, the artist selects a surface area using an external 3D modeling tool, we used Autodesk Maya. This set of surface vertices is then mapped to the nearest vertices in the tetrahedral mesh (Fig. 8). All the tetrahedral vertices included between the selected vertices and the nearest bone are also selected. The average distance $\overline{d}$ from the nearest bone is then computed. If the distance of a vertex from the nearest bone is smaller than $\overline{d}$, the motion of the vertex is governed by both LBS and PBD. If the distance is greater than $\overline{d}$, only PBD is applied in that area. The artist can change interactively the stiffness of all the constraints influenced by these latter vertices, reducing or increasing the jiggling effect until the desired effect is reached.

5. Results

We tested our method on a variety of articulated characters of different shapes, polygonal resolution and topology. The tetrahedral meshes were generated using [BO05], the *walking* and the *kick* motion capture data was available from the Carnegie Mellon University Motion Capture DataBase [GS01], and the remaining ones were hand-made. We implemented our method in C++ and timings for various animation sequences were measured on a mass-market laptop equipped with an Intel i5 2.50 GHz processor and 4GB RAM. To advance the dynamics, we used a 10 ms time step and 12 iterations per frame. Mean computation times are reported in Table 1. Even by using a fairly high number of constraints (e.g., the BIGBUNNYHQ test case, ~34K constraints), our system is able to compute the animation of the tetrahedral mesh and map the deformation on the render mesh at interactive rate.

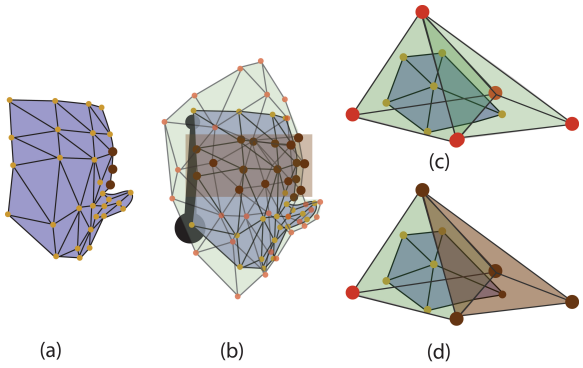


Figure 8: Soft selection mechanism. (a) Surface mesh, including the selected vertices. (b) Surface mesh embedded in the tetrahedral mesh, the tetrahedron vertices placed between the selected surface vertices and the nearest bone are selected automatically. (c) A close-up view of two tetrahedrons with the embedded surface composed by six vertices. (d) The surface vertex is selected by the artist, and the nearest vertices on the tetrahedron are selected automatically. All of these vertices are shown in brown and the brown bar highlighting these vertices.

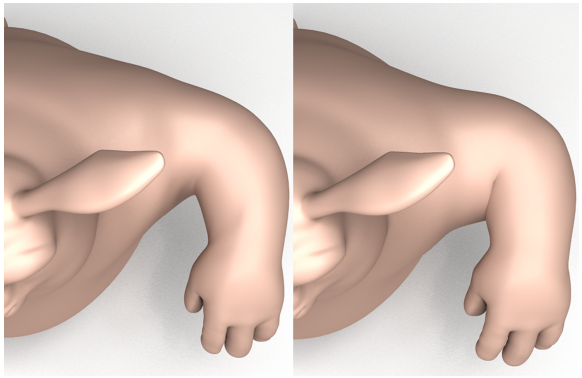


Figure 9: Comparison with and without volume preservation. The volume of skin is preserved by solving the PBD constraints. *Left.* Step 1: LBS is applied. Note the loss of volume at the elbow joint. *Right.* Step 2: Positions of the vertices are adjusted solving the PBD constraints (stretch, volume and bind), preserving the volume and avoiding the *candy-wrapper* effect.

Figs. 1 and 11 show the TORSO, the HORSE and the BIG-BUNNY models animated with a walking cycle. All of these models exhibit large jiggling effects in the area of the belly. Fig. 13 shows an octopus model with articulated tentacles and jiggling effects in the head zone. Fig. 10 provides a visual comparison between the deformations achieved with our method, plain Linear Blend Skinning (LBS) and Dual Quaternion Skinning (DQS). Our system is not affected either by the *candy-wrapper* effect or by undesired skin bulging. Fig. 12 shows the HUMAN model performing a highly dynamic motion sequence involving large joint rotations, which demonstrates the robustness of our system. Please refer to the enclosed video for the animated results.

To assess the accuracy of our method, we compute the relative error w.r.t. to the volume of the character's mesh: $e = V/V_0$, where V is the volume of the deformed mesh at the current frame, and V_0 is the volume of the mesh in the initial rest position. As shown from the quantities reported in Table 1, our method successfully preserves the volume of the character during the animation ($e \leq 0.5\%$). Fig. 9 shows an example of the deformation of the arm with and without volume conservation.

6. Limitations and Future Work

Currently, the skinning weights in our method are computed according to a simple heuristic formula (Eq. 2). Those weights are simple to construct, but may not be optimal for any given input mesh. Alternatively, the skinning weights can be computed using the techniques in [DdL13], which uses voxels to approximate the geodesic distance for computing bone influence weights. In future work, we would like to explore the possibility of applying their algorithm in the first step of our method to bolster the second step and produce higher quality smooth bindings.

Our implementation does not detect and resolve self-collisions, which may lead to geometry overlapping (Fig. 14), and therefore our method cannot guarantee self-intersection-free deformations. In the future we would like to extend our framework to support temporary constraints for implementing self-collisions. Furthermore, we would like to explore the use of localized PBD in areas like the extremities of the limbs where our method, based on global PBD, may produce undesired effects.

7. Conclusion

We have presented a simple and fast skinning algorithm for skeleton-driven deformations of articulated characters. During the animation, the deformation model preserves the volume and allows for passive jiggling behavior. Our system initializes the blend weights and the soft constraints automatically, so it does not require a considerable set-up effort. Artists can define specific areas of the body and decide on the amount of jiggling affecting them by tuning a single scalar stiffness parameter.

In contrast to other methods (e.g., [KP11, DB13]), our system does not model the inner structure of the human skin. However, using a combination of widely known and relatively simple techniques, Linear Blend Skinning and Position-Based Dynamics, we achieved believable animations with a time performance suitable for interactive appli-

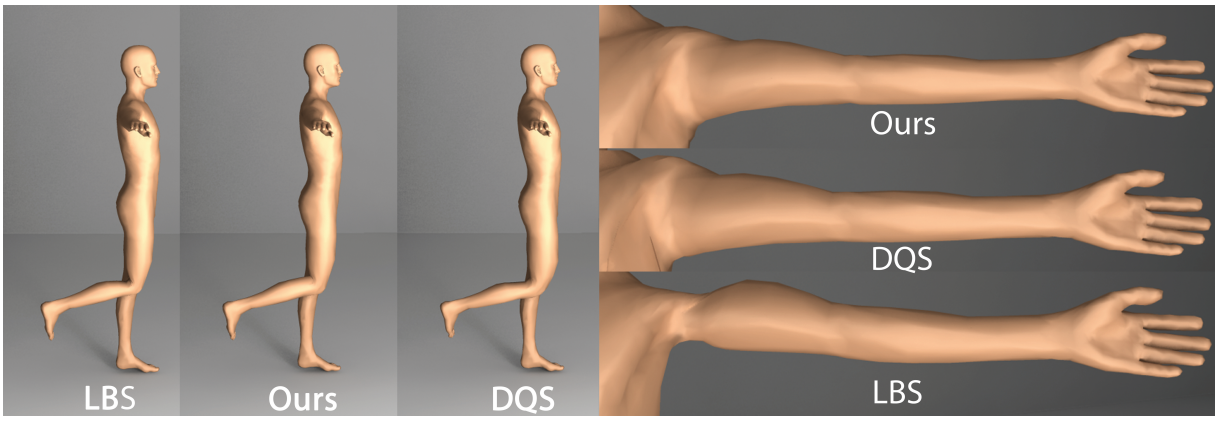


Figure 10: Our method does not suffer from the *candy-wrapper* artefacts of linear blend skinning (LBS) and the bulging artefacts of dual quaternion skinning (DQS).

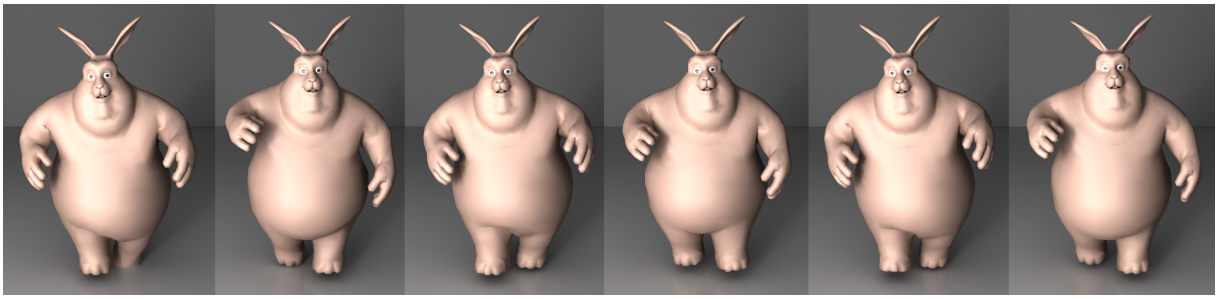


Figure 11: Realistic jiggling motion in the belly area for the BIGBUNNY character at interactive rates (34k constraints, 71 fps).

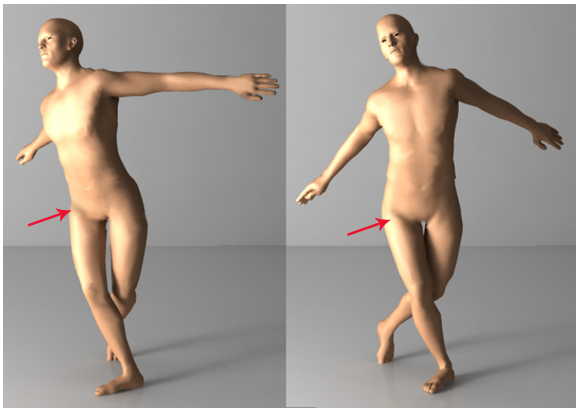


Figure 14: Self-collisions are not explicitly handled. This may lead to geometry overlapping in the contact regions (note the dark regions near the articulations).

cations. In the first step, the character is deformed by applying Linear Blend Skinning, then in the second step the position of the vertices is accommodated using a Position-Based Dynamics solver. The first step in our skinning method, the LBS deformer, improves the convergence speed of the PBD solver, while maintaining the elastic nature of the character body. We employ a graph coloring algorithm for parallelizing the computation of the geometrical constraints. This leads to fast performances even in case of a fairly high number of tetrahedrals. We mapped the tetrahedral mesh to the input skin geometry for achieving high quality renderings.

Being based on Position-Based Dynamics, the elastic behavior of the soft body deformer is influenced by the number of iterations employed in the parallel Gauss-Seidel solver. We employed 12 iterations during our tests, but in general the artist has to heuristically choose a value which depends on the topology and the polygonal resolution of the input mesh.

8. Acknowledgements

The authors would like to thank the anonymous reviewers for their insightful comments and suggestions. We are grateful to the Avempace Erasmus Mundus Project for funding postgraduate scholarship to Nadine Abu Rumman.

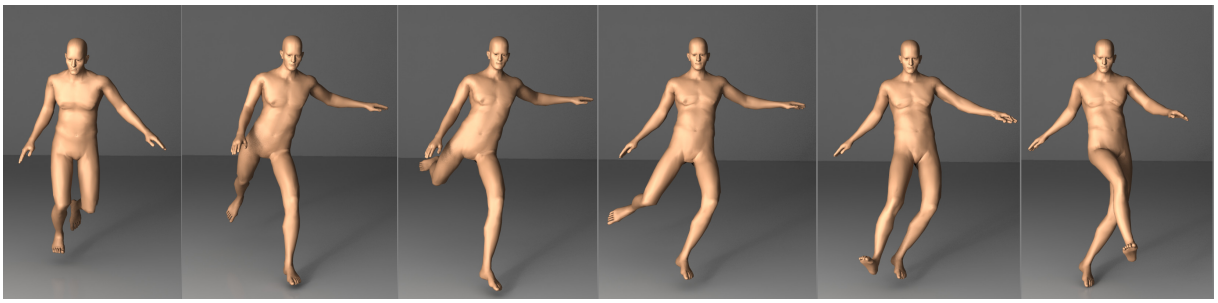


Figure 12: Highly dynamic sequence, HUMAN character (12k constraints, 146 fps).

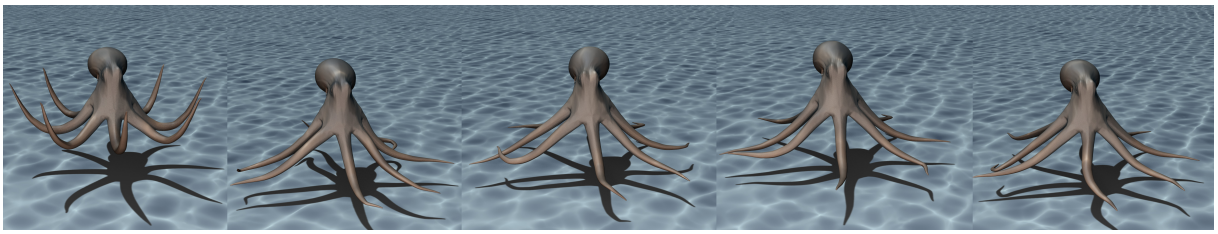


Figure 13: Our method automatically produces believable skin deformations of the soft tissues for an octopus moving at interactive rates (14k constraints, 145.6 fps).

References

[APH11] ALDRICH G., PINSKIY D., HAMANN B.: Collision-Driven Volumetric Deformation on the GPU. In *In Proceedings of Eurographics 2011, Short Papers* (Llandudno, UK, 2011), Eurographics Association, pp. 9–12. 3

[AS07] ANGELIDIS A., SINGH K.: Kinodynamic skinning using volume-preserving deformations. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2007, San Diego, California, USA, August 2-4, 2007* (2007), pp. 129–140. 2

[BMO*14] BENDER J., MÜLLER M., OTADUY M. A., TESCHNER M., MACKLIN M.: A survey on position-based simulation methods in computer graphics. *Computer Graphics Forum 33*, 6 (2014), 228–251. 4, 6

[BMOT13] BENDER J., MÜLLER M., OTADUY M. A., TESCHNER M.: Position-based methods for the simulation of solid objects in computer graphics. In *EUROGRAPHICS 2013 State of the Art Reports* (Girona, Spain, 2013), Eurographics Association. 3

[BO05] BOISSONNAT J.-D., OUDOT S.: Provably good sampling and meshing of surfaces. *Graph. Models 67*, 5 (2005), 405–451. 3, 6

[BP07] BARAN I., POPOVIĆ J.: Automatic rigging and animation of 3d characters. In *ACM SIGGRAPH 2007 Papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. 2

[CA005] Dynamic skinning: Adding real-time dynamic effects to an existing character animation. In *Proceedings of the 21st Spring Conference on Computer Graphics* (New York, NY, USA, 2005), SCCG '05, ACM, pp. 87–93. 3

[CBC*05] CAPELL S., BURKHART M., CURLESS B., DUCHAMP T., POPOVIĆ Z.: Physically based rigging for deformable characters. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2005), SCA '05, ACM, pp. 301–310. 3

[CGC*02] CAPELL S., GREEN S., CURLESS B., DUCHAMP T., POPOVIĆ Z.: Interactive skeleton-driven dynamic deformations. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 586–593. 3

[CHP89] CHADWICK J. E., HAUMANN D. R., PARENT R. E.: Layered construction for deformable animated characters. *SIGGRAPH Computer Graphics 23*, 3 (1989), 243–252. 3

[CM83] COLEMAN T. F., MORE J. J.: Estimation of sparse jacobian matrices and graph coloring problems. *Journal of Numerical Analysis 20* (1983), 187–209. 6

[DB13] DEUL C., BENDER J.: Physically-based character skinning. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (Lille, France, Nov. 2013), Eurographics Association. 7

[DdL13] DIONNE O., DE LASA M.: Geodesic voxel binding for production character meshes. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2013), SCA '13, ACM, pp. 173–180. 2, 7

[FO06] FORSTMANN S., OHYA J.: Fast Skeletal Animation by skinned Arc-Spline based Deformation . In *In Proceedings of Eurographics 2006, Short Papers* (Vienna, Austria, 2006), pp. 1–4. 2

[Fra12] FRATARCANGELI M.: Position-based facial animation synthesis. *Computer Animation and Virtual Worlds 23*, 3-4 (2012), 457–466. 3

[GOT*07] GALOPPO N., OTADUY M. A., TEKIN S., GROSS

M. H., LIN M. C.: Soft articulated characters with fast contact handling. *Comput. Graph. Forum 26*, 3 (2007), 243–253. 3

[GS01] GROSS R., SHI J.: *The CMU Motion of Body (MoBo) Database*. Tech. Rep. CMU-RI-TR-01-18, Robotics Institute, Carnegie Mellon University, 2001. 6

[HTC*13] HAHN F., THOMASZEWSKI B., COROS S., SUMNER R., GROSS M.: Efficient simulation of secondary motion in rig-space. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Anaheim, California, 2013), SCA '13. 1

[JBPS11] JACOBSON A., BARAN I., POPOVIĆ J., SORKINE O.: Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. 30*, 4 (July 2011), 78:1–78:8. 2

[JRMG07] JACKA D., REID A., MERRY B., GAIN J.: A comparison of linear skinning techniques for character animation. In *Proceedings of the 5th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa* (New York, NY, USA, 2007), AFRIGRAPH '07, ACM, pp. 177–186. 2

[JT05] JAMES D. L., TWIGG C. D.: Skinning mesh animations. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 399–407. 2

[JZvdP*08] JU T., ZHOU Q.-Y., VAN DE PANNE M., COHEN-OR D., NEUMANN U.: Reusable skinning templates using cage-based deformations. *ACM Trans. Graph. 27*, 5 (Dec. 2008), 122:1–122:10. 2

[KCZO07] KAVAN L., COLLINS S., ZÁRA J., O'SULLIVAN C.: Skinning with dual quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2007), I3D '07, ACM, pp. 39–46. 2

[KH14] KIM Y., HAN J.: Bulging-free dual quaternion skinning. *Journal of Visualization and Computer Animation 25*, 3-4 (2014), 323–331. 3

[KJ11] KIM T., JAMES D. L.: Physics-based character skinning using multi-domain subspace deformations. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, ACM, pp. 63–72. 3

[KJP02] KRY P. G., JAMES D. L., PAI D. K.: Eigenskin: Real time large deformation character skinning in hardware. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2002), SCA '02, ACM, pp. 153–159. 2

[KP11] KIM J., POLLARD N. S.: Fast simulation of skeleton-driven deformable body characters. *ACM Trans. Graph. 30*, 5 (Oct. 2011), 121:1–121:19. 3, 7

[KS12] KAVAN L., SORKINE O.: Elasticity-inspired deformers for character articulation. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 196:1–196:8. 3

[LCF00] LEWIS J. P., CORDNER M., FONG N.: Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 165–172. 2

[LKCOL07] LIPMAN Y., KOPF J., COHEN-OR D., LEVIN D.: Gpu-assisted positive mean value coordinates for mesh deformations. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, 2007), SGP '07, Eurographics Association, pp. 117–123. 2

[LTW95] LEE Y., TERZOPOULOS D., WATERS K.: Realistic modeling for facial animation. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1995), SIGGRAPH '95, ACM, pp. 55–62. 3

[LYWG13] LIU L., YIN K., WANG B., GUO B.: Simulation and control of skeleton-driven soft body characters. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 215:1–215:8. 1

[MDM*02] MÜLLER M., DORSEY J., MCMILLAN L., JAGNOW R., CUTLER B.: Stable real-time deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2002), ACM, pp. 49–54. 3

[MG03] MOHR A., GLEICHER M.: Building efficient, accurate character skins from examples. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 562–568. 2

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (Apr. 2007), 109–118. 2, 4

[MMG06] MERRY B., MARAIS P., GAIN J.: Animation space: A truly linear framework for character animation. *ACM Trans. Graph. 25*, 4 (Oct. 2006), 1400–1423. 2, 4

[MTLT88] MAGNENAT-THALMANN N., LAPERRIÈRE R., THALMANN D.: Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics Interface '88* (Toronto, Ont., Canada, 1988), Canadian Information Processing Society, pp. 26–33. 2

[MZS*11] MCADAMS A., ZHU Y., SELLE A., EMPEY M., TAMSTORF R., TERAN J., SIFAKIS E.: Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph. 30*, 4 (July 2011), 37:1–37:12. 1, 3

[NMK*06] NEALEN A., MUELLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum 25*, 4 (Dec. 2006), 809–836. 3

[PH08] PARK S. I., HODGINS J. K.: Data-driven modeling of skin and muscle deformation. In *ACM SIGGRAPH 2008 Papers* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 96:1–96:6. 2

[PSE03] POPOVIĆ J., SEITZ S. M., ERDMANN M.: Motion sketching for control of rigid-body simulations. *ACM Trans. Graph. 22*, 4 (Oct. 2003), 1034–1054. 3

[RLN06] RHEE T., LEWIS J. P., NEUMANN U.: Real-time weighted pose-space deformation on the GPU. *Computer Graphics Forum 25*, 3 (2006), 439–448. 2

[SK00] SINGH K., KOKKEVIS E.: Skinning characters using surface-oriented free-form deformations. In *Canadian Human-Computer Communications Society, Graphics Interface 2000* (Montreal, Quebec, Canada, 2000), pp. 35–42. 2

[SZT*08] SHI X., ZHOU K., TONG Y., DESBRUN M., BAO H., GUO B.: Example-based dynamic skinning in real time. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 29:1–29:8. 3

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. *SIGGRAPH Computer Graphics 21*, 4 (Aug. 1987), 205–214. 3

[TT93] TURNER R., THALMANN D.: The elastic surface layer model for animated character construction. In *Proceedings on of computer graphics international '93* (Lausanne,Switzerland, 1993), Springer-Verlag Berlin and Heidelberg GmbH and Co. K, pp. 399–412. 3

[VBG*13] VAILLANT R., BARTHE L., GUENNEBAUD G., CANI

M.-P., ROHMER D., WYVILL B., GOURMEL O., PAULIN M.: Implicit skinning: Real-time skin deformation with contact modeling. *ACM Trans. Graph. 32*, 4 (July 2013), 125:1–125:12. 2

[vFTS06] VON FUNCK W., THEISEL H., SEIDEL H.-P.: Vector field based shape deformations. *ACM Trans. Graph. 25*, 3 (2006), 1118–1125. 2

[WP02] WANG X. C., PHILLIPS C.: Multi-weight enveloping: Least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2002), SCA '02, ACM, pp. 129–138. 2

Model	# vertices	# tetra	# bones	S	T	B	CT_{volume} [ms]	$CT_{stretch}$ [ms]	CT_{bind} [ms]	CT_{total} [ms]	# iterations	volume loss %	$k_{stretch}$	fps
HUMAN	9528	4998	25	6342	4998	1654	2.1	3.2	1.507	6.807	12	0.09	0.9	146.9
HUMANHQ	9528	10516	25	13223	10516	2645	3.43	5.113	1.627	10.17	12	0.12	0.9	98.3
BIGBUNNY	8111	7827	29	9732	7827	2108	2.71	3.74	1.53	7.98	12	0.27	0.6	125.3
BIGBUNNYHQ	8111	14022	29	17367	14022	3190	5.41	6.56	1.95	13.92	12	0.39	0.6	71.8
OCTOPUS	4000	4339	63	7617	4339	1667	2.08	3.28	1.507	6.867	12	0.45	0.6	145.6
HORSE	3833	6126	43	9437	6126	1826	2.65	3.52	1.52	7.69	12	0.17	0.8	130
TORSO	4345	4280	25	6076	4280	1074	2.07	3.17	1.469	6.709	12	0.15	0.6	149.1

Table 1: Skinning performance. #vertices: number of initial vertices in the render mesh, #tetra: number of elements in the tetrahedral mesh, S: number of stretch constraints, T: number of volume constraints, B: number of bind constraints, fps: avg. frame rate, CT: avg. skinning computation time during 1 sec simulation, where $(CT_{total} = CT_{volume} + CT_{stretch} + CT_{bind})$.