Lattice-based Skinning and Deformation for Real-time Skeleton-driven Animation

Cheng-Hao Chen, I-Chen Lin, Ming-Han Tsai, Pin-Hua Lu
Dept. of Computer Science,
National Chiao Tung University,
Hsinchu City, Taiwan
email: {cch@caig.cs, ichenlin@cs, ParkerTsai@caig.cs, sailors@caig.cs}.nctu.edu.tw

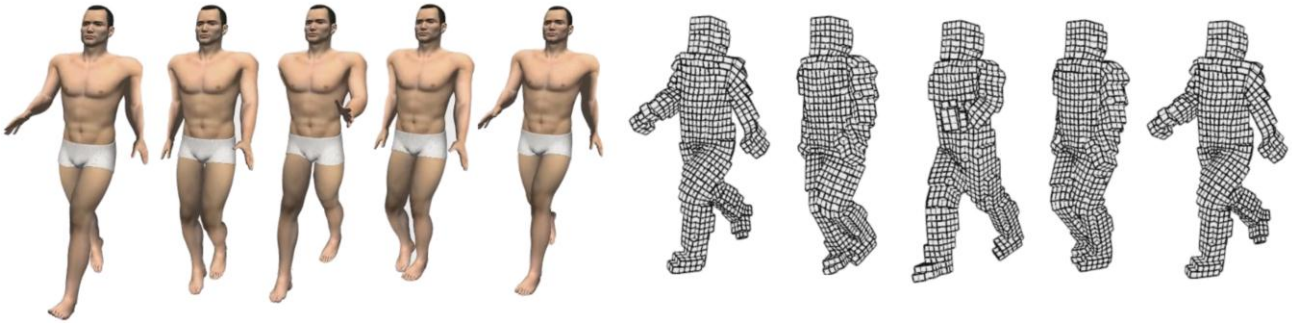


Figure 1. Skeleton-driven animation with primary and secondary deformation. Left: the rendered character surface; right: the lattice structure (cells) for efficient deformation computation.

***Abstract*—In this paper, we present an efficient framework to deform polygonal models for skeleton-driven animation. Standard solutions of skeleton-driven animation, such as linear blend skinning, require intensive artist intervention and focus on primary deformations. The proposed approach can generate both low- and high-frequency surface motions such as muscle deformation and vibrations with little user intervention. Given a surface mesh, we construct a lattice of cubic cells embracing the mesh and we apply lattice-based smooth skinning to drive the surface primary deformation with volume preservation. Lattice shape matching with dynamic particles, in the meantime, is utilized for secondary deformations. Due to the highly parallel lattice structure, the proposed method is liable to GPU computation. Our results show that it is adequate to vividly real-time animation.**

Keywords-skeleton-driven animation; secondary deformation; skinning

I. INTRODUCTION

Skinning and skeleton-driven animation are the technologies behind character animation and are widely used in video games or movie production. Skinning models define how geometric surfaces change according to a function of the skeletal poses. Skinning can be modeled in a data-driven style by example-based data regression to estimate the shape for a new pose [20]. It can also be modeled procedurally in the case of physically-based or anatomy-based approaches.

A popularly-used method is called: Linear Blend Skinning (LBS), as known as Skeletal Subspace Deformation (SSD) [12]. Besides human skin, it can also be applied to clothes and other deformable surfaces [3]. The principle of LBS is to represent transformations of vertices as linearly-blended matrices. This method produces artifacts such as "candy-wrapper" effects in the deformed surface. In spite of such shortcomings, linear blending is still the most popular skinning approach due to its simplicity and efficiency.

On the other hand, physics-based simulated skin deformation can produce surface bulging, jiggle of fat tissues and other dynamic effects. However, many skinning or deformation approaches are often devoid of such secondary deformation effects [15] or has to utilize a separate simulation component. But it increases the difficulty in structure switching and parameter-tuning for both skinning and secondary deformations.

Our system takes a unified framework, where skinning, secondary deformation and volume preservation are mainly evaluated through regular 3D grids and their vertices, called cells and particles, respectively. The effects then distribute to vertices of polygonal models.

After automatically evaluating the deformable parts and skinning weights through a heat-propagation-like method, our system estimates primary deformation by linear blend skinning on all particles. Secondary deformation is then generated by extending the lattice shape matching (LSM) method [16] to every particle in the cells. In the original lattice shape matching method, increasing the shape matching region causes the rigidity. In our case, the shape matching region size is related to the smoothness of mesh. Since the cell volume may not be preserved during deformation, especially those near the joints, we propose hierarchically preserving volume through all joint-dependent deformable parts.

The shape matching regions and deformable parts are automatically computed and can be manually adjusted as well. The mesh can be partially soft or rigid according to the shape matching regions and mesh parameters. These material properties can even be changed dynamically. Fig. 1 shows

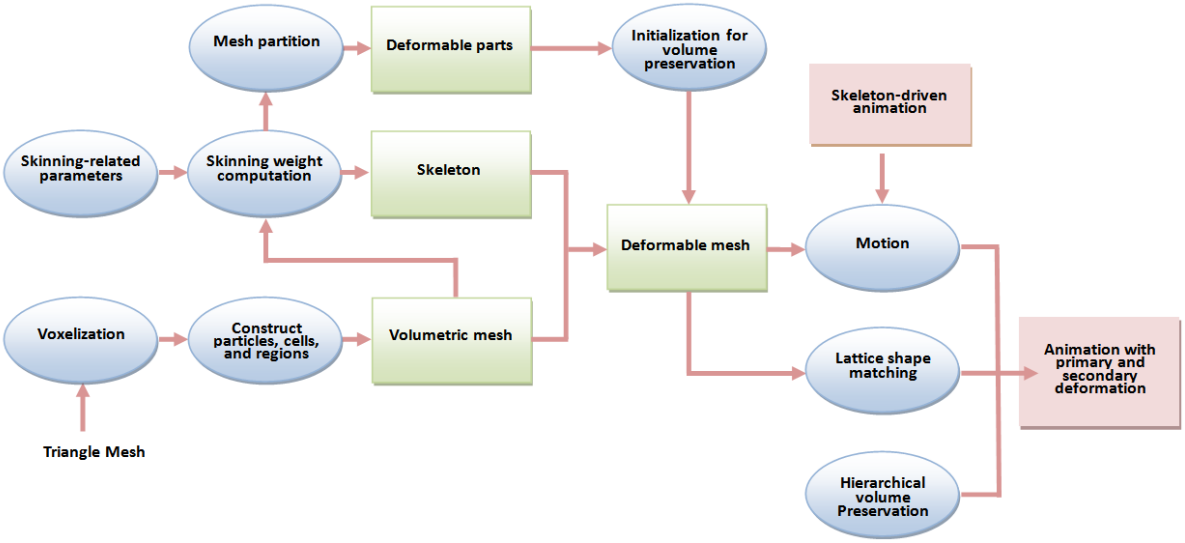


Figure 2. The flowchart of the proposed system.

our skeleton-driven animation, where skinning and enhanced secondary deformation on the chest are applied. Fig. 2 shows the flowchart of the proposed system. Our main contribution includes:

- A unified and efficient framework for combing skinning, volume preservation and secondary volume deformation.
- Lattice-based skinning method with automatic skinning weight computation.
- A hierarchical volume preservation technique that can reduce "*candy-wrapper*" effect.

II. Related Work

Skinning techniques are widely used to drive realistic animated characters. Many significant improvements of linear blend skinning are implemented with a variety of compromises between user control, skinning effort, storage requirements, and performance. Pose Space Deformation [11] addressed well-known artifacts like collapsing joints. Dual Quaternion Skinning [10] introduces effective rotation-based interpolations. Wang et al. [20] proposed a rotational regression method to capture advanced skin deformation such as muscle bulging, and twisting. Zhou et al. [22] proposed Volumetric Graph Laplacian (VGL) to deform the mesh based on 2D curves. All of above methods focus on the primary deformation of the surface mesh.

Shi et al. [17] proposed an example-based approach with surface detail preservation and secondary deformations. However, example-based methods usually require expensive manual works or data acquisition. Von Funck et al. [19] added elastic secondary deformation to a given primary deformation by a small number of user-placed mass-spring sets.

Forstmann et al. proposed alleviating skinning artifacts based on auxiliary curved skeletons [5], but it increased complexity of the GPU implementation and inconsistency with the established skinning pipeline. Lattice-based shape deformations are widely used to animate embedded geometry [4]. Regular voxel [6, 13] or body-centered cubic tetrahedral meshes [14] can simplify meshing issues for simulation. Other research [7, 8, 9] deformed a character using a simpler mesh, and are mainly used for direct manipulation.

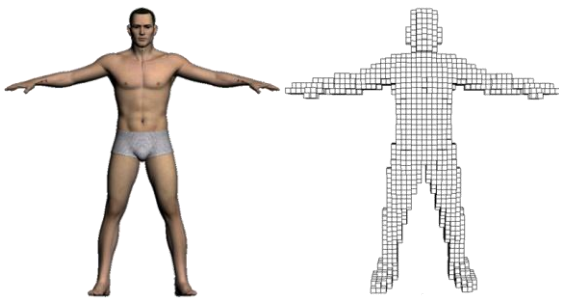


Figure 3. The initial pose for voxelization.

III. Lattice-based Smooth Skinning

We now define the lattice representation of mesh, and show how to apply smooth skinning method on the mesh.

A. Lattice construction

Given a target surface mesh, we voxelize the mesh to construct a lattice of cubic cells containing the mesh [6]. The surface mesh should be in an appropriate initial pose as show in Fig. 3. The voxelization level can be adjusted by users according to the detail of input models and animation. The embedded mesh can now be deformed by trilinear interpolation of eight particles (cell vertices) positions. Let $\boldsymbol{P}$ denote the set of all particles. For each particle p in $\boldsymbol{P}$, we denote its static initial position as x_p^0, its dynamic position

as x_p, and its mass as m_p. Each particle has its index represented by a 3-tuple related to the reference (or origin) particle. $p_{(x, y, z)}$ denotes a particle with index (x, y, z). We set neighbor $\boldsymbol{N_p}$ as a set of particles that stay within 3x3x3 cells surrounding p. We also define the adjacency of p as a set of particles that have one cell distance away from p.

B. Smooth skinning

In this subsection, we describe our smooth skinning method on the voxelized mesh (cells). Deforming a model with skinning techniques requires a skeleton structure, the skin and skinning vertex weights. The skin is a 3D triangular mesh without assumption on connectivity. The skeleton is a rooted tree, where the nodes represent joints and the edges can be interpreted as bones. Fig. 4 shows the surface mesh, skeleton of a target character. In our implementation, we provide user interfaces for assigning rough skeleton nodes and our system then approaches these nodes to local volume centers. On the other hands, automatic skeleton extraction is mentioned in related articles [21].

Without loss of generality, transformations of joints and skeletons in each hierarchical level are assumed to be rigid. In the classic skinning framework [12], the vertex weights describe the skin-to-skeleton binding (*i.e.*, the amount of influence of individual joints on each vertex). In our case, we first consider particle weights instead of weights for surface vertices. Assume that there are k joints in the input skeleton. Each joint has an associated local coordinate system in its initial position. The transformation from the initial position of joint $j \in \{j_1, \ldots, j_k\}$ to its current position can be expressed by a rigid transformation matrix – $\boldsymbol{T_j} \in SE(3)$. We assume that particle p is attached to joints $j_p = \{j_1, \ldots, j_n\}$ with weights $w_p = \{w_p^{\,1}, \ldots, w_p^{\,n}\}$. The indices $j_1, \ldots, j_n$ are integers referring to the joints that influence a given particle; $w_p^{\,i}$ represents the influence of joint j_i on particle p. Most skinning applications let n to be four due to graphics hardware considerations (we store j_p in a *vec4*-typed variable in GLSL). The weights are normally assumed to be convex and $\sum_{i=1}^{n} w_i = 1$ and $w_i \geq 0$. The particle positions x_p deformed by linear blend skinning is then computed as:

$$x_p' = \sum_{i=1}^{n} w_p^j T_{j_i} x_p = \left(\sum_{i=1}^{n} w_p^j T_{j_i} \right) x_p , \qquad (1)$$

where $\boldsymbol{T_{ji}}$ is the transformations of joint i. The blended matrix $\sum_{i=1}^{n} w_p^i T_{ji}$ is not guaranteed to be a rigid transformation, even if all $\boldsymbol{T_{ji}}$ are rigid. To overcome this problem, these transformations $\boldsymbol{T_{ji}}$ are factorized into rotation $\boldsymbol{R_{ji}}$ and scale/shear $\boldsymbol{S_{ji}}$ components by using the polar decomposition

$$T_{j_i} = R_{j_i} S_{j_i} . \qquad (2)$$

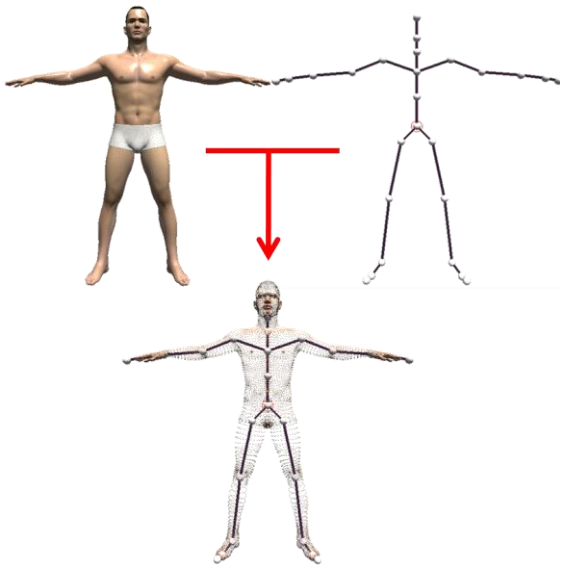


Figure 4. The input character mesh, skeleton and mesh-skeleton mapping.

We use the fast polar decomposition technique described in [16] and build a new transformation $\dot{T}_{j_i}$ to replace $\boldsymbol{T_{ji}}$ by $\boldsymbol{R_{ji}}$.

C. Particle skinning weight assignment

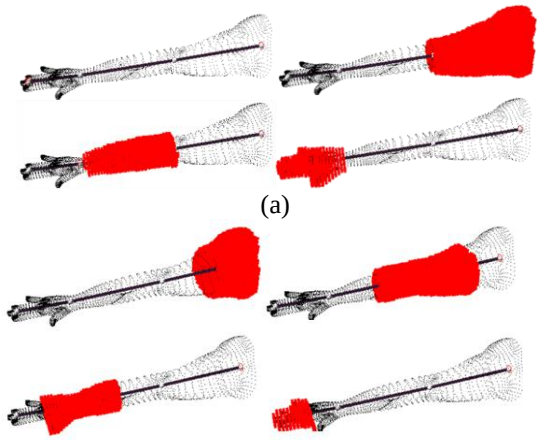
Skinning weights are usually specified by artists according to bone size and joint influence regions. A recent technique proposed as an automatic algorithm for unguided skeleton mesh called bone heat [1]. This method aims at extracting the skeleton and weight through a heat diffusion system on the surface of the mesh. The heat diffusion mechanism is more efficient and reasonable on our regular volumetric mesh than the original volume approximation on thin shields.

First, we treat each bone j as a heat source with energy e_j influenced by user-specified parameters such as bone width or bone length. For each heat source j, we compute the directly-influenced particles $\hat{P}_j$ which are the closest particles to the bone j by no more than one cell size. The energy of particles in $\hat{P}_j$ is assumed to be e_j. Then we construct an undirected simple graph G**:**

$$G = (V, E), V = P$$

$$E = \left\{ (p_1, p_2) \mid p_1 \in P, p_2 \in P, p_2 \in N_{p_1} \right\}. \qquad (3)$$

Each particle is considered as a node in G, and having edges with its neighbors. Let $cost(p_i, p_j)$ denote the cost of edge (p_i, p_j), p_i has an index (x_i, y_i, z_i), and p_j has an index (x_j, y_j, z_j). The edge cost is proportional to the Euclidean distance. When we apply heat diffusion from the directly-



(a)

(b)

Figure 5. The deformable parts (red part). (a) bone-dependent deformable part. (b) joint-dependent deformable part that combine two adjacent bone-dependent parts in halves.

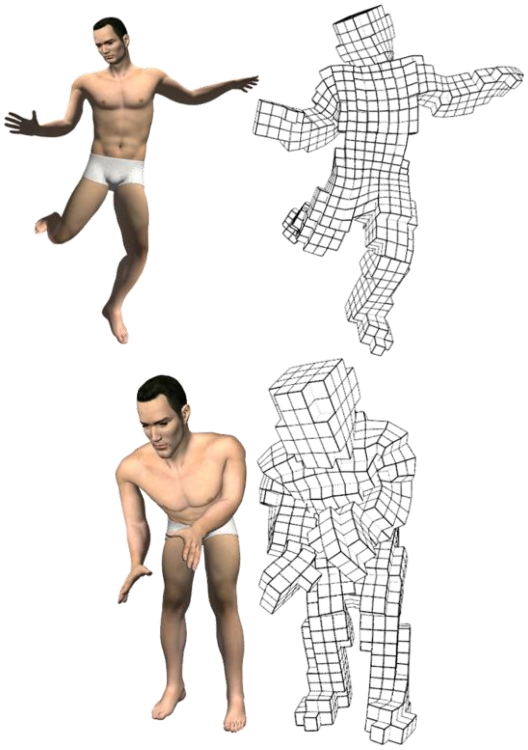


Figure 6. Lattice-based smooth skinning.

influenced particles $\hat{P}_j$ to all other particles, a particle's energy is a weighted combination of the edge cost to its neighbors. The neighbors' energy to particle p_i is denoted by $ew_{(p_i,p_j)}$, and computed as:

$$ew_{(p_i,p_j)} = \frac{wt(p_i,p_j)}{\sum_{p_k \in N_{p_i}} wt(p_i,p_k)}$$

$$wt(p_i,p_j) = \frac{b_i}{cost(p_i,p_j)+1}, \tag{4}$$

where b_i is bone strength that influences the energy attenuation. Hence, a particle's energy is then computed as:

$$e_{p_i} = \sum_{p_k \in N_{p_i}} ew_{(p_i,p_k)} e_{p_k}. \tag{5}$$

The diffusion runs repeatedly until the completion of diffusion process. After the heat diffusion, we partition all particles into several deformable parts based on the most effective bone to each vertex as in Fig. 5(a). For each bone-dependent deformable part, we divide the particles into two subparts and combine two subparts adhering to the same joint to form joint-dependent deformable parts as shown in Fig. 5(b). The joint deformable parts are basic units for our hierarchical volume preservation. Besides automatic segmentation and weights, we also allow users to adjust the attributes.

Skeletal motions can now be used to drive cell particles with skinning. Accordingly, the vertices on polygonal models are moved through interpolation. Fig. 6 presents a skinned human model using our lattice-based smooth skinning method. Since the concept of our lattice-based skinning approach is similar to the basic linear blend skinning, its performance is almost as efficient as linear blend skinning.

D. Volume preservation

Traditional linear blend skinning has deformation artifacts such as "candy-wrapper" effect since it does not address the unnatural volume changes. We present a hierarchical approach extended from the method by Takamatsu and Kanai [18]. First, we define:

$$\begin{aligned} diff_p &= \left\{ \frac{1}{2}\left(x_q^0 - x_p^0\right) | q \in Adj_p \right\} \\ &= \left\{ d_p^{x+}, d_p^{x-}, d_p^{y+}, d_p^{y-}, d_p^{z+}, d_p^{z-} \right\} \end{aligned}, \tag{6}$$

where $diff_p$ denotes the set of half distance from p to its six adjacencies. Then, the volume of each particle p can be defined as:

$$\begin{aligned} &Vol(p) = \\ &d_p^{x+} \cdot \left(d_p^{y+} \times d_p^{z+}\right) + d_p^{x+} \cdot \left(d_p^{y-} \times d_p^{z-}\right) \\ &+ d_p^{x-} \cdot \left(d_p^{y+} \times d_p^{z-}\right) + d_p^{x-} \cdot \left(d_p^{y-} \times d_p^{z+}\right) \\ &+ d_p^{z+} \cdot \left(d_p^{y+} \times d_p^{x-}\right) + d_p^{z+} \cdot \left(d_p^{y-} \times d_p^{x+}\right) \cdot \\ &+ d_p^{z-} \cdot \left(d_p^{y+} \times d_p^{x+}\right) + d_p^{z-} \cdot \left(d_p^{y-} \times d_p^{x-}\right) \end{aligned} \tag{7}$$

The operator **.** means dot product and the × means cross product. The volume of mesh is then computed as:

$$Vol(P) = \sum_{p \in P} Vol(p) \quad (8)$$

After mesh deformation, all particles ***P*** transform to their new positions. Let ***P'*** be the deformed particles. We define the particle displacement field:

$$\begin{aligned} \hat{V} &= \{\hat{v}_1, \ldots, \hat{v}_{|P|}\} \\ &= \{s_1 u_1 R_1, \ldots, s_{|P|} u_{|P|} R_{|P|}\}, \end{aligned} \quad (9)$$

where $\{R_1, ..., R_{|P|}\}$ are a set of particle's rotations. $\{u_1, ..., u_{|P|}\}$ are particles' outward vectors which point to the nearest boundary. $\{s_1, ..., s_{|P|}\}$ are particle's volume correction scales. They are inverse related to the nearest Manhattan distance to the boundary. This means a particle closer to the boundary has a larger percentage to keep the local volume consistent.

With the displacement field, we can evaluate how each particle should be adjusted to keep the part volume the same through the following equation:

$$Vol(P) = Vol(P' + \lambda \hat{V}), \quad (10)$$

where λ is the unknown value.

We correct the positions of particles from the root joint-dependent deformable part to its all sub-parts. For skeletal-driven animation, our approach preserving local volumes is more adequate than that for global volume [18]. For instance, deforming the left arm should not significantly influence the volume on the legs. Compared with deformation methods preserving cell rigidity and volume by optimization [2], the proposed method is relatively light-weight in computing, since the evaluation of particle outward vectors and scales are deterministic and applicable to parallel computation.

IV. Skinning with Secondary Deformation

In this section, we introduce how we combine our lattice-based skinning method with the lattice shape matching to generate secondary deformation. It can make the body part "soft" if there is less skeleton binding.

A. Lattice shape matching

In the previous section, we construct a lattice of cubic cells containing the surface mesh. Now we further define shape matching region for each particle. Each particle p is associated with a shape matching region comprised of a set of shape matching particles, $Region_p$. A $Region_p$ of half-width $\hat{w}$ contains p and all particles reachable within a Manhattan distance $\hat{w}$ from particle p. For instance, if $\hat{w} = 1$, $Region_p = N_p$.

The main lattice shape matching algorithm is proposed by Rivers and James [16]. At each time step, each $Region_r$ finds the best rigid transformation $\tilde{T}_r$ by least-squares to match the initial particle positions x_p^0 to their deformed positions x_p for $p \in Region_r$. Therefore, each particle p's goal position g_p can be calculated by average regional rigid transformation of the particle's position:

$$g_p = \frac{1}{|Region_p|} \left(\sum_{r \in Region_p} \tilde{T}_r \right) x_p^0. \quad (11)$$

To generate the secondary deformation, we establish a dynamic system according to differences between the particle position x_p and the goal position g_p and the external force f_p, as shown in (12) and (13)

$$v_p(t+h) = v_p(t) + \alpha \frac{g_p(t) - x_p(t)}{h^2} + h \frac{f_p(t)}{m_p} \quad (12)$$

$$x_p(t+h) = x_p(t) + h v_p(t+h), \quad (13)$$

where h is the simulation time step, $x_p(t)$ and $v_p(t)$ are the position and velocity at t, respectively.

Applying dynamics calculation to all particles results in "gummy bear" like deformation. To embed skeleton in to the cells, we further assign the particles within bone cylinders to be rigidly adhered on the bone. The effect of "bone rigidness" for other particles depends on the region windows. In general, those closer to the bone axis can have more rigidness.

B. Combination of Skinning and Lattice Shape Matching

Both the skinning and the shape matching update particle positions. In order to generate secondary deformation by lattice shape matching with guidance of skinning, we use the result of (1)(2) and particle p's dynamic position x_p to obtain

Figure 7. Skinning artifacts. Upper: with lower voxelization resolution (1072 particles); lower: with higher voxelization

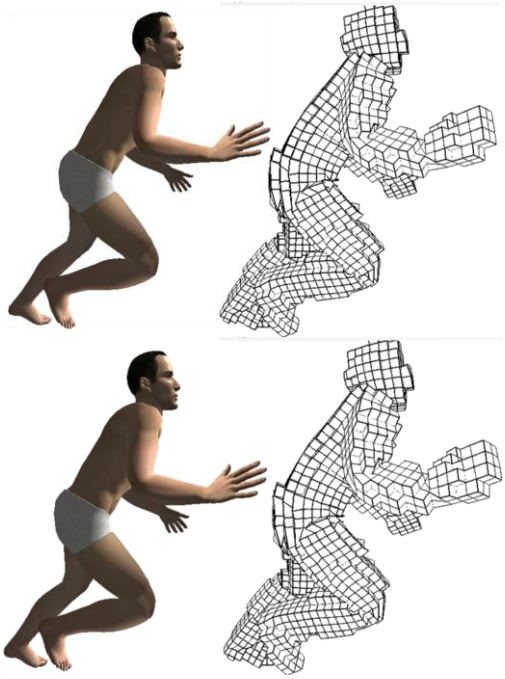


Figure 8. The deformation result. Upper: without hierarchical volume preservation. Lower: with hierarchical volume preservation. The volumes near the joints are closer to the original ones.

$$\hat{x}_p = \delta x_p + (1-\delta)\overline{x}_p, \tag{14}$$

where δ is the ratio of secondary deformation and $0 \leqq \delta \leqq 1$. If δ = 0, the result is same as lattice-based smooth skinning. The larger δ is applied, the more obvious the secondary deformation appears. Users can freely adjust δ or even switch different δ profiles during simulation to obtain more realistic effects. $\hat{x}_p$ is then applied to the lattice shape matching process for the best transformation as described in section 4.1.

At each time step, each particle p vibrates between x_p and $\overline{x}_p$. The goal position g_p will be more and more close to $\overline{x}_p$. x_p will gradually converge toward $\overline{x}_p$. Besides, we also include the damping force described in [16] to speed up the convergence.

TABEL I. PERFORMANCE TEST ON A MODERATE LAPTOP AND A DESKTOP.

# of cells	# of particles	Average Frame Per Second(laptop)	Average Frame Per Second(desktop)
524	1072	148.572	215.294
976	1906	92.717	134.181
1241	2451	61.684	99.436
1905	3058	42.376	81.633
Triangle Mesh: 28059 vertices, 55888 triangles. Time step: 16ms. # of joints: 38 Region size: 2		CPU: Intel Core 2 Duo P8600 RAM:DDR3-1066 4GB Display: Nvidia GeForce G105M	CPU: Intel Core 2 Quad Q6600 RAM:DDR2-800 8GB Display: Nvidia GeForce 8800GT

V. EXPERIMENT AND RESULT

Our approach is flexible since we provide an interactive environment and various adjustable mesh parameters with defaults for users. Fig. 3 shows our subject mesh, skeleton, and mesh-skeleton mapping. The skeleton motion data we used are from CMU's motion capture database [23] and a 30Hz dataset for interactive applications mentioned in [24]. Fig. 7 shows the results of voxelization resolution test. An improper resolution (1092 particles) results in skinning artifacts. For our test surface mesh (28059 vertices), we choose the resolution about 1900 particles to get balance between skinning quality and performance.

Fig. 8 show the results of volume correction. The volume preservation method is capable of alleviating the "*candy-wrapper*" effect resulting from linear blend skinning. Table 1 show performance tests for various voxelization resolutions on a laptop and desktop.

VI. CONCLUSION

We present a lattice-based framework for surface deformation in skeleton-driven animation and editing. After voxelizing and mapping an input surface mesh with cells, our system automatically generates lattice-based skinning weights through diffusion-based influence propagation. The

skinning deformation is then combined with dynamic particles of lattice shape matching to approximate the physically-realistic secondary deformation. To reduce the deformation artifacts, a hierarchical method for local volume preservation is employed.

The proposed system requires only skeleton-driven motion data and triangle mesh as inputs, and it can generate appealing animation with little user intervention. The experiment shows that our efficient designs make our system adequate to real-time computation even on a moderate laptop computer.

REFERENCE

[1] I. Baran and J. Popović, "Automatic rigging and animation of 3d characters", ACM Trans. Graph., vol. 26, no. 3, article: 72, 2007.

[2] M. Botsch, M. Pauly, M. Wicke, and M. Gross, "Adaptive Space Deformations Based on Rigid Cells", Computer Graphics Forum, vol. 26, no.3, pp.339-347, 2007.

[3] F. Cordier and N. Magnenat-Thalmann, "A data-driven approach for real-time clothes simulation", Proc. Pacific Graphics, pp. 257–266, 2004.

[4] P. Faloutsos, M. van de Panne, and D. Terzopoulos, "Dynamic free-form deformations for animation synthesis", IEEE Trans. Visualization and Computer Graphics, vol. 3, no. 3, pp. 201–214, 1997.

[5] S. Forstmann, J. Ohya, A. Krohn-Grimberghe, and R. McDougall, "Deformation styles for spline-based skeletal animation", Proc. 2007 ACM SIGGRAPH / Eurographics symposium on Computer animation, pp. 141–150, 2007.

[6] D. L. James, J. Barbič, and C. D. Twigg, "Squashing cubes: Automating deformable model construction for graphics", Proc. ACM SIGGRAPH 2004 Conference, Sketches & Applications, 2004.

[7] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki, "Harmonic coordinates for character articulation", ACM Trans. Graph., vol. 26, no. 3, article: 71, 2007.

[8] T. Ju, S. Schaefer, and J. Warren, "Mean value coordinates for closed triangular meshes", ACM Trans. Graph., vol. 24, no. 3, pp. 561–566, 2005.

[9] T. Ju, Q. Zhou, M. van de Panne, D. Cohen-Or, and U. Neumann, "Reusable skinning templates using cage-based deformations", ACM Trans. Graph., vol. 27, no. 5, article: 122, 2008.

[10] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan, "Geometric skinning with approximate dual quaternion blending", ACM Trans. Graph., vol. 27, no. 4, article: 105, 2008.

[11] J. P. Lewis, M. Cordner, and N. Fong, "Pose space deformation: a unified approach to shape interpolation and skeletondriven deformation", Proc. ACM SIGGRAPH'00, pp. 165–172, 2000.

[12] N. Magnenat-Thalmann, R. Laperri`ere, and D. Thalmann, "Joint-dependent local deformations for hand animation and object grasping", Proc. Graphics Interface'88, pp. 26–33, 1988.

[13] M. Müller and M. Gross, "Interactive virtual materials", Proc. Graphics Interface'04, pp. 239–246, 2004.

[14] N. Molino, Z. Bao, and R. Fedkiw, "A virtual node algorithm for changing mesh topology during simulation", ACM Trans. Graph., vol. 23, no. 3, pp. 385–392, 2004.

[15] J. F. O'Brien, V. B. Zordan, and J. K. Hodgins, "Combining active and passive simulations for secondary motion", IEEE Computer Graphics and Applications, vol. 20, no.4, pp. 86–96, 2000.

[16] A. R. Rivers and D. L. James, "Fastlsm: fast lattice shape matching for robust realtime deformation", ACM Trans. Graph., vol. 26, no. 3, article: 82, 2007.

[17] X. Shi, K. Zhou, Y. Tong, M. Desbrun, H. Bao, and B. Guo, "Example-based dynamic skinning in real time", ACM Trans. Graph., vol. 27, no.3 article: 29, 2008.

[18] K. Takamatsu and T. Kanai, "Volume-preserving lsm deformations", Proc. ACM SIGGRAPH ASIA 2009, Sketches, article: 15, 2009.

[19] W. von Funck, H. Theisel, and H.-P. Seidel, "Elastic secondary deformations by vector field integration", Proc. Eurographics Symposium on Geometry Processing'07, pp. 99–108, 2007.

[20] R. Y. Wang, K. Pulli, and J. Popović, "Real-time enveloping with rotational regression", ACM Trans. Graph., vol. 26, no.3 artist: 55, July 2007.

[21] Y.-S. Wang, and T.-Y. Lee, "Curve-Skeleton Extraction Using Iterative Least Squares Optimization", IEEE Trans. Visualization and Computer Graphics, vol.14, no.4, pp. 926-936, 2008.

[22] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H. Shum, "Large mesh deformation using the volumetric graph laplacian", ACM Trans. Graph., vol.24, no.3, pp. 496–503, 2005.

[23] CMU GraphicsLab, Motion Capture Database. http://mocap.cs.cmu.edu.

[24] I.-C. Lin, J.-Y. Peng, C.-C. Lin, M.-H. Tsai, "Adaptive Motion Data Representation with Repeated Motion Analysis", IEEE Trans. Visualization and Computer Graphics, vol.17, no. 4, pp.527-538, April, 2011.