

Skeleton-driven surface deformation through lattices for real-time character animation

Cheng-Hao Chen · Ming-Han Tsai · I-Chen Lin ·
Pin-Hua Lu

Published online: 1 November 2012
© Springer-Verlag Berlin Heidelberg 2012

Abstract In this paper, an efficient deformation framework is presented for skeleton-driven polygonal characters. Standard solutions, such as linear blend skinning, focus on primary deformations and require intensive user adjustment. We propose constructing a lattice of cubic cells embracing the input surface mesh. Based on the lattice, our system automatically propagates smooth skinning weights from bones to drive the surface primary deformation, and it rectifies the over-compressed regions by volume preservation. The secondary deformation is, in the meanwhile, generated by the lattice shape matching with dynamic particles. The proposed framework can generate both low- and high-frequency surface motions such as muscle deformation and vibrations with few user interventions. Our results demonstrate that the proposed lattice-based method is liable to GPU computation, and it is adequate to real-time character animation.

Keywords Lattice-based shape · Character skinning · Secondary deformation · Skeleton-driven animation

1 Introduction

Character animations are now extensively used in video games and movie production. To make the characters more realistic, skeleton-driven animation with skinning deformation is one of the most efficient methods in real-time applications. Skeletal character animation represents the basis of character motion by a hierarchical bone structure.

The bone movement can be evaluated by motion capture devices [31], motion synthesis techniques [13, 22] or manual indication, e.g. forward or inverse kinematics. Skinning models, on the other hand, define how geometric surfaces transform according to distinct bones. Skinning can be modeled in an example-based style by data regression to estimate the shape for a new pose [12, 28]. It can also be modeled procedurally in physics-based or anatomy-based approaches.

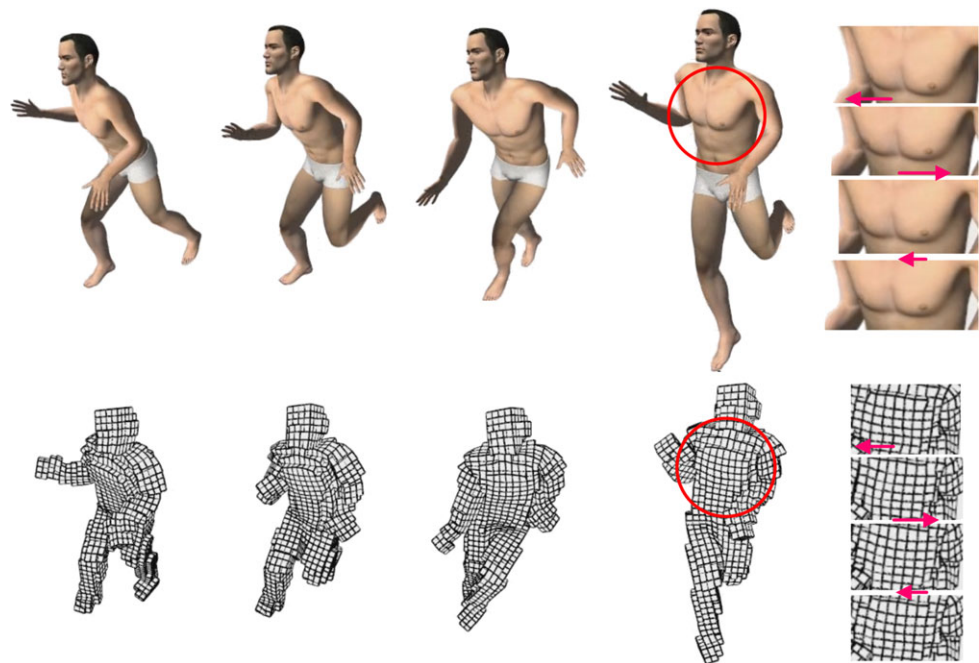
A well-known method is called Linear Blend Skinning (LBS), also known as Skeletal Subspace Deformation (SSD) [17]. The principle of LBS is to represent transformations of vertices as linearly blended matrices. The blending weights of vertices are usually indicated by skillful artists. Even with careful weight tuning, this kind of method may still produce artifacts, such as joint collapsing or candy-wrapper effects, on the deformed surface. However, due to its simplicity and computational efficiency, linear blending is still the most popular skinning approach. Besides human skin, LBS can also be applied to clothes or other deformable surfaces [4].

On the other hand, simulation-based skin deformation can produce surface bulging, jiggle of fat tissues and other dynamic phenomena. However, many skinning or deformation approaches are devoid of such secondary deformation effects [21]; otherwise, they have to utilize a separate simulation component. Such a strategy increases the overhead in data correspondence and parameter-tuning between skinning and secondary deformation structures.

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

C.-H. Chen · M.-H. Tsai · I.-C. Lin (✉) · P.-H. Lu
Department of Computer Science, National Chiao Tung
University, 1001 Ta-Hsueh Road, Hsinchu City, 30010, Taiwan
e-mail: ichenlin@cs.nctu.edu.tw

Fig. 1 Skeleton-driven animation with primary and secondary deformation. *Upper*: the rendered character surface; *lower*: the lattice structure (cells) for efficient deformation computation. The *conceptual arrows* show the on-going moving directions



Given an input mesh and its skeleton, we automatically evaluate the deformable parts and skinning weights by a heat-propagation-like method. The primary deformation is evaluated by linear blend skinning on particles and cells. While we further simulate the dynamic movement of particles, we can then generate surfaces with secondary deformation by extending the lattice shape matching (LSM) method [23].

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. Moreover, since a part of cell volume may be over-compressed by weighted blending, we present a hierarchically preserving volume through all joint-dependent deformable parts.

In our system, the shape matching regions and deformable parts are automatically computed and allow manual adjustment as well. The deformation of polygonal model can be partially soft or rigid according to the shape matching regions and mesh parameters. These material properties can even be changed dynamically. Figure 1 shows our skeleton-driven animation, where skinning and enhanced secondary deformation on the chest are applied. Please browse our demo video for details. Our main contribution includes:

- A unified and efficient framework for combing skinning, volume preservation and secondary volume deformation.
- Automatic skinning weight computation by a lattice-based propagation method.
- A hierarchical volume preservation technique that can alleviate the collapsing effect.

2 Related work

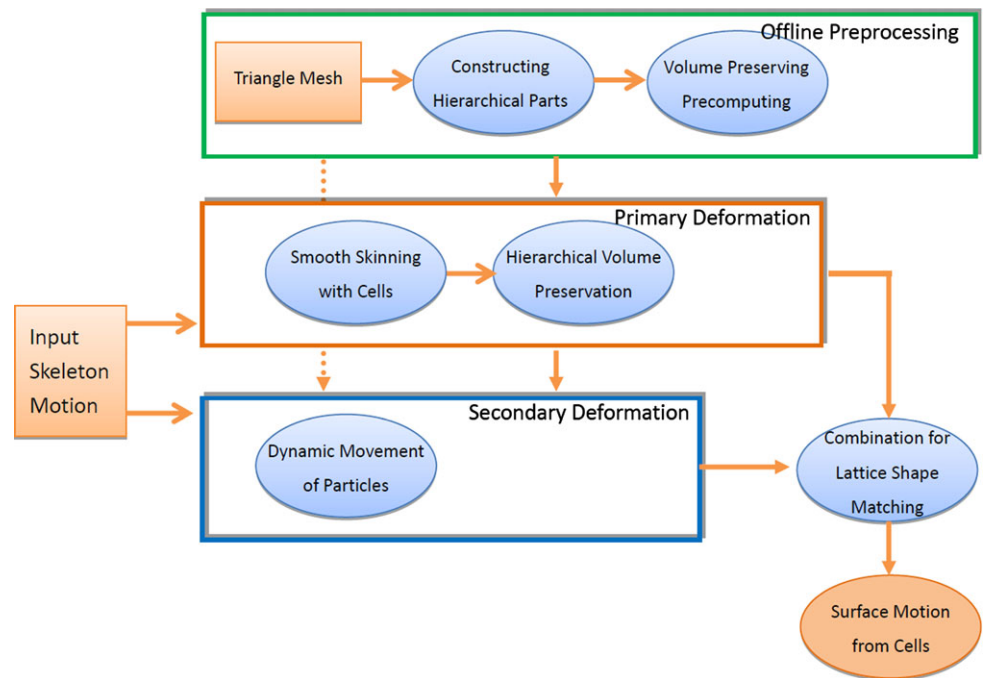
Skinning techniques are widely used to drive realistic deformable characters. Various modifications of linear blend skinning are proposed for different purposes, such as user control, skinning effort, storage requirements, or computational performance.

Pose Space Deformation [14] addressed well-known collapsing of joint artifacts. Dual Quaternion Skinning [11] introduces effective rotation-based interpolations. Wang et al. [28] proposed a rotational regression method to capture advanced skin deformation such as muscle bulging, and twisting. Zhou et al. [30] proposed Volumetric Graph Laplacian (VGL) to deform the mesh based on 2D curves. The above-mentioned methods focused on the primary deformation of the surface mesh.

Shi et al. [24] proposed an example-based approach with surface detail preservation and secondary deformations. However, example-based methods usually require intensive manual adjustment or data acquisition. Von Funck et al. [27] 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 [6], 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 [5]. Regular voxel [7, 19] or body-centered cubic tetrahedral meshes [20] can simplify meshing issues for simulation. Other research [8–10] deformed a character using a simpler mesh, and are mainly

Fig. 2 The flowchart of the proposed system. The *upper part* describes the offline preprocess; the *lower part* demonstrates the online stages



used for direct manipulation. Nevertheless, simulating detailed volumetric deformation is expensive for many real-time applications.

Detailed lattice-based FEM meshes are used in character animation [25], but mostly for offline simulations. It is possible to avoid recomputing elements by using rotated linear element models [3, 18], but integrating large-deformation dynamics often involves significant computation for large linear systems. Besides, the patch-based method [32], using a fewer number of B-spline control points with springs, efficiently performed surface deformation.

Nevertheless, without carefully design, the volume of deformation body near joints, such as elbows and knees, may change drastically. It usually causes serious collapses, such as the joint collapsing effect. To fix the problem, Takamatsu et al. [26] restricted the control point’s position by introducing displacement fields, while Hyun et al. [33] tried to preserve the volume of body parts with sweep surfaces.

3 System overview

Figure 2 shows the flowchart of the proposed system. Given a surface mesh to be deformed, we conservatively voxelize the mesh to construct a lattice of cubic cells containing the mesh. In the later operations, such as the skinning and dynamic evaluations, we apply these computations on particles of cells instead of mesh vertices. Our method would automatically compute per-particle skinning weights through volumetric energy diffusion. After the skinning weight computation, we group all particles into several sets called deformable parts based on the joints and bones of the applied

skeleton. For each joint-dependent deformable part, we evaluate the hierarchical volumes to preserve their volume during online deformation. Properties such as rigidity, smoothness, and intensity of secondary deformation, can be adjusted by users as well.

For online process, the input skeletal motions drive the skinning deformation on cells. The over-compressed volumes of deformable parts are then rectified by volume preservation. The dynamic movements are also controlled by unit point-mass particles placed on the cell corners. Each particle is associated with a shape matching region comprising a set of shape matching particles. We apply rigid shape matching transforms around every particle for smooth and robust deformation without domain boundary artifacts. After combining the primary and secondary cell deformation, the embedded vertices within a cell are then deformed by trilinear interpolation of all particles’ positions.

We implement our efficient method on multiple-threading and GPU. We organize all particles’ data as a linear texture and stored in GPU’s global memory, and then use a trilinear lattice deformer to move the surface mesh. By the computation power of GPUs, the performance can be substantially improved for real-time applications.

4 Lattice-based skinning

In this section, we define the lattice representation for the surface mesh, and show how to apply smooth skinning on the mesh.

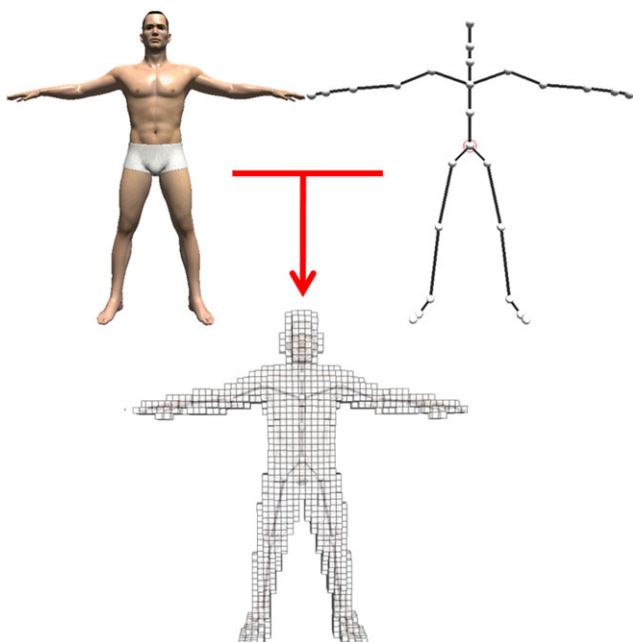


Fig. 3 The input character mesh, skeleton and cell-skeleton mapping

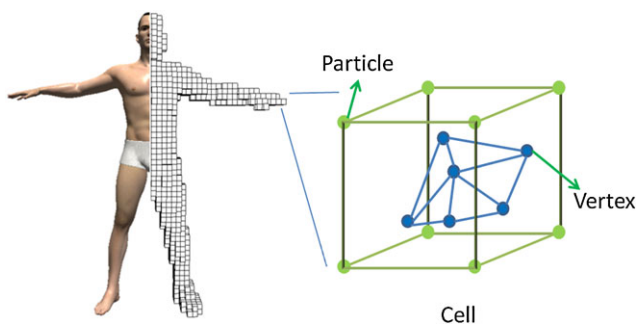


Fig. 4 The cell structure of a target character

4.1 Lattice construction

Given an input surface mesh, we voxelize the mesh to construct a lattice of cubic cells containing the mesh [7]. The surface mesh should be in an appropriate initial pose as shown in Fig. 3. Users can adjust the voxelization level according to the details of input models and animation. The embedded mesh can now be deformed by trilinear interpolation of eight particles (cell vertices) positions.

Figure 4 shows the cell structure. Let \mathbf{P} denote the set of all particles. For each particle p in \mathbf{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 denote neighbor N_p as a set of particles locating within $3 \times 3 \times 3$ cells surrounding p . We also define the adjacency of p as a set of particles in one cell distance away from p . The particle re-

lations are evaluated and recorded during the initialization stage.

4.2 Smooth skinning with voxels

This subsection describes smooth skinning on the voxelized mesh (cells). Deforming a model with skinning techniques requires a skeleton structure, the skin and the skinning weight for each vertex. 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 represent the bones. Figure 3 shows the surface mesh, skeleton of a target character. In our current implementation, we provide interfaces for users to assign rough skeleton nodes, and our system then approaches these nodes to local volume centers. This manual initialization process can be replaced by automatic skeleton extraction described in related articles [15, 29].

Without loss of generality, transformations of individual joints and bones are assumed to be rigid. In the classic skinning framework [17], 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 from its initial posture. For a joint $j \in \{j_1, \dots, j_k\}$, the transformation from the initial position to its current position can be expressed by a rigid transformation matrix— $T_j \in SE(3)$.

We assume that particle p is attached to joints $j_p = \{j_1, \dots, j_n\}$ with weights $w_p = \{w_p^1, \dots, w_p^n\}$. The integer indices of j_1, \dots, j_n refer to the joints that influence a given particle p ; w_p^i represents the influence of joint j_i on particle p . Most skinning applications set n to 4 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 $\sum_{i=1}^n w_p^i = 1$ and $w_p^i \geq 0$. The particle positions x_p deformed by linear blend skinning is then computed as

$$x'_p = \sum_{i=1}^n w_p^i T_{j_i} x_p = \left(\sum_{i=1}^n w_p^i T_{j_i} \right) x_p \tag{1}$$

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

$$T_{j_i} = R_{j_i} S_{j_i} \tag{2}$$

We use the fast polar decomposition technique described in [23] and build a new transformation \hat{T}_{j_i} to replace T_{j_i} by R_{j_i} .

4.3 Automatic skinning weight estimation

Conventionally, skinning weights are specified by artists according to bone size and joint influence regions. A recent technique, called bone heat [1], automatically estimated weights for unguided skeleton mesh. This method aims at extracting the skeleton and weight through a heat diffusion system on the surface of the mesh. Nevertheless, the heat propagation on thin shields is different from solid volume; authors also rectified this problem by volume approximation on the shields. In our case, we approximate the propagation on regular cells instead of thin shields. The heat diffusion mechanism is more computationally efficient and reasonable.

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 within one cell width. The energy of particles \hat{P}_j are assumed to be e_j . Then we construct an undirected simple graph G :

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

$$E = \{(p_1, p_2) | p_1 \in P, p_2 \in P, p_2 \in N_{p_1}\} \quad (3)$$

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 influenced particles \hat{P}_j to all other particles, the particles' energy transformation rates are dependent on the edge cost to its neighbors. A neighbor's energy transfer rate to particle p_i is denoted by $ew_{(p_i, p_j)}$, and computed as

$$ew_{(p_i, p_j)} = \frac{wt(p_i, p_j)}{\sum_{p_k \in N_{p_i}} wt(p_i, p_k)} \quad (4)$$

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

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

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

The diffusion runs repeatedly until the completion of diffusion process. To obtain the particle weights, we compare the energy from all heat sources for each particle and then normalize them to ensure the weights are convex. Skeletal motions can now be used to drive cell particles with skinning. Accordingly, the vertices on polygonal models are moved through interpolation. Figure 5 presents a skinned human model using our lattice-based smooth skinning method. Since the concept of our lattice-based skinning

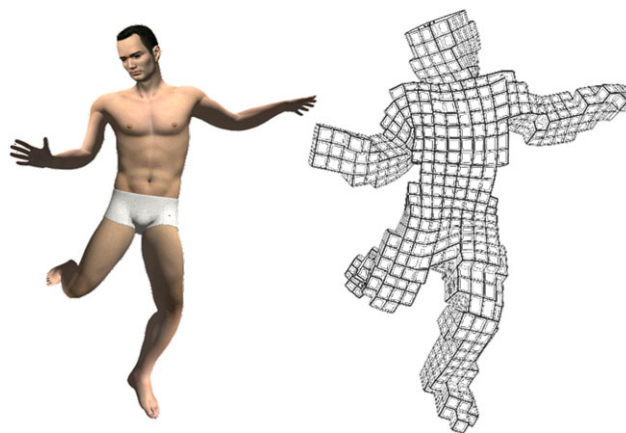


Fig. 5 Lattice-based smooth skinning. Left: the rendered character surface; right: the lattice structure (cells)

approach is similar to the basic linear blend skinning, its performance is almost as efficient as linear blend skinning.

4.4 Hierarchical volume preservation

Traditional linear blend skinning has deformation artifacts, such as joint collapsing, since it does not consider unnatural volume changes. We present a hierarchical approach to approximately preserve the volume.

First of all, we have to group unit cells into higher-level parts. Since we have calculated the bone influences on cells, we further make use of this information, and group all particles into several deformable parts based on their most effective bone as in Fig. 6a. Besides bone-dependent deformable parts, we can also divide the part particles into two subparts and combine two subparts adhering to the same joint as joint-dependent deformable parts. As shown in Fig. 6b, the joint deformable parts are basic units for our hierarchical volume preservation.

To evaluate the volume of deformable parts, we extend the global volume preservation by Takamatsu and Kanai [26]. We denote $diff_p$ as a set of vectors whose lengths are a half distance from p to its six adjacencies.

$$diff_p = \left\{ \frac{1}{2}(x_q^0 - x_p^0) | q \in Adj_p \right\}$$

$$= \{d_p^{x+}, d_p^{x-}, d_p^{y+}, d_p^{y-}, d_p^{z+}, d_p^{z-}\} \quad (6)$$

Then, the volume of each particle p can be defined based on the $diff_p$ vectors:

$$Vol(p) = d_p^{x+} \cdot (d_p^{y+} \times d_p^{z+}) + d_p^{x-} \cdot (d_p^{y-} \times d_p^{z-})$$

$$+ d_p^{x-} \cdot (d_p^{y+} \times d_p^{z-}) + d_p^{x+} \cdot (d_p^{y-} \times d_p^{z+})$$

$$+ d_p^{z+} \cdot (d_p^{x+} \times d_p^{y-}) + d_p^{z+} \cdot (d_p^{x-} \times d_p^{y+})$$

$$+ d_p^{z-} \cdot (d_p^{x+} \times d_p^{y+}) + d_p^{z-} \cdot (d_p^{x-} \times d_p^{y-}) \quad (7)$$

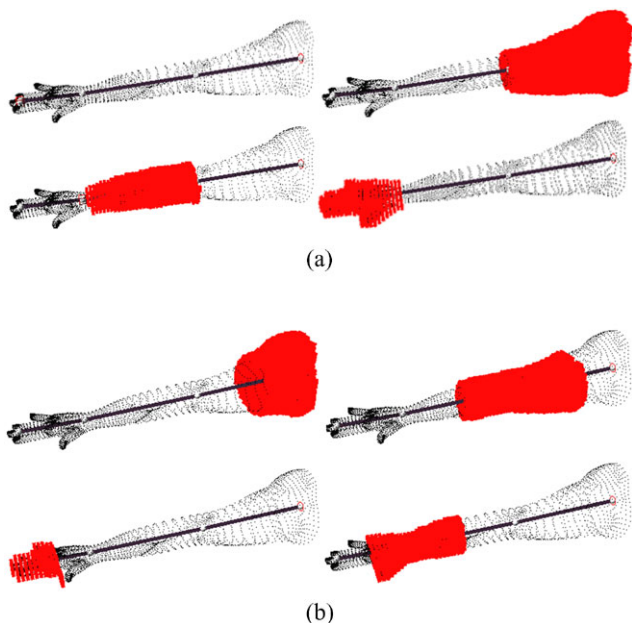


Fig. 6 The deformable parts (*red part*). (a) Bone-dependent deformable part. (b) Joint-dependent deformable part that combines two adjacent bone-dependent parts in halves

The operator \cdot and \times represent the dot and cross product, respectively. The volume of a deformable part is then defined as the total volume of its particles:

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

After mesh deformation, all particles P transform to their new positions. Let P' be the deformed particles. The particle displacement field is

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

where $|P|$ represents the particle amount; $\{R_1, \dots, R_{|P|}\}$ are a set of particle's rotations; $\{u_1, \dots, u_{|P|}\}$ are the particles' outward vectors which point toward the nearest boundary. $\{s_1, \dots, s_{|P|}\}$ are the particle's volume correction scales. These scales are inverse related to the nearest Manhattan distance to the boundary. This means that a particle closer to the boundary has a larger percentage to keep the local volume consistent. We choose this strategy because we observe that the deformations usually change larger on the surface than the internal regions. We are inclined to accentuate the surface contributions in volume preservation. Figure 7 shows the composition of a displacement field V . u controls the direction of pulling each vertex, and s controls the percentages of a particle in volume changes.

With the displacement field, now we can rectify the current deformed mesh volume $Vol(P')$ to the desired one. Since the volume should be preserved, we evaluate how each

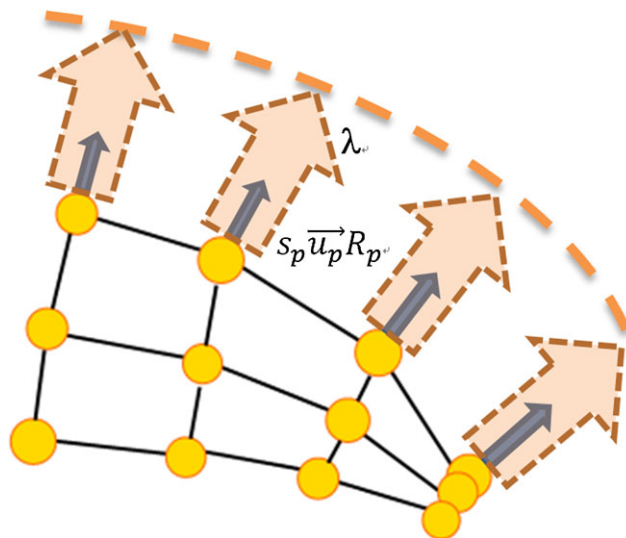


Fig. 7 Components of displacement field. For a particle p , its displacement vector results from $s_p u_p R_p$. An appropriate λ is evaluated to preserve the volume after deformation

particle should be adjusted to keep the part's volume the same through the following equation:

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

where λ is the unknown value. We make use of the cubic equation solution proposed by Takamatsu and Kanai [26] to solve the unknown λ .

$$\begin{aligned} Vol(P' + \lambda \hat{V}) &= \sum_{p \in P'} Vol(p + \lambda \hat{v}_p) \\ &= \sum_{p \in P'} (a_p^0 + a_p^1 \lambda + a_p^2 \lambda^2 + a_p^3 \lambda^3) \end{aligned} \tag{11}$$

where

$$\begin{aligned} a_p^0 &= \sum_p m_p^x \cdot (m_p^y \times m_p^z) \\ a_p^1 &= \sum_p n_p^x \cdot (m_p^y \times m_p^z) + \sum_p m_p^x \cdot (n_p^y \times m_p^z) \\ &\quad + \sum_p m_p^x \cdot (m_p^y \times n_p^z) \\ a_p^2 &= \sum_p m_p^x \cdot (n_p^y \times n_p^z) + \sum_p n_p^x \cdot (m_p^y \times n_p^z) \\ &\quad + \sum_p n_p^x \cdot (n_p^y \times m_p^z) \\ a_p^3 &= \sum_p n_p^x \cdot (n_p^y \times n_p^z) \end{aligned}$$

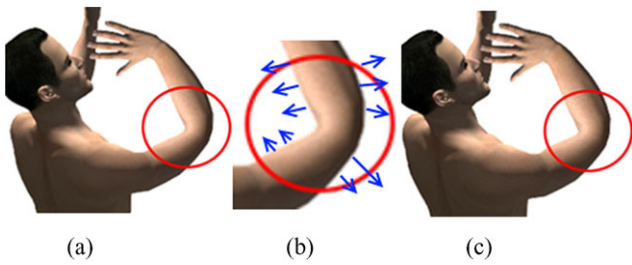


Fig. 8 Volume preservation example: (a) without volume preservation; (b) blue arrows show the displacement field; (c) with volume preservation

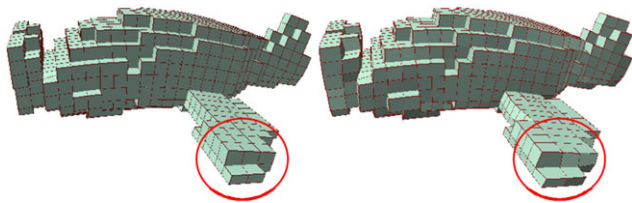


Fig. 9 A mesh pushed by a downward external force. *Left:* without volume preservation. *Right:* with volume preservation

Similar to Eq. (6), m_p and n_p are difference vectors for new position x'_p and current displacement field \hat{v}_p .

$$\begin{aligned} & \left\{ \frac{1}{2}(x'_q - x'_p) | q \in Adj_p \right\} \\ &= \{m_p^{x+}, m_p^{x-}, m_p^{y+}, m_p^{y-}, m_p^{z+}, m_p^{z-}\} \\ & \left\{ \frac{1}{2}(\hat{v}_q - \hat{v}_p) | q \in Adj_p \right\} \\ &= \{n_p^{x+}, n_p^{x-}, n_p^{y+}, n_p^{y-}, n_p^{z+}, n_p^{z-}\} \end{aligned} \tag{12}$$

Figure 8 shows an example of preserving volume near elbow. The muscle size near the elbow is enlarged through the displacement field. We apply volume preservation on each joint-dependent deformable part instead of the whole mesh, such as [26]. An example is shown in Fig. 9. While a force push downward, we correct the positions of particles from the root joint-dependent deformable part to all its child parts.

For skeletal-driven animation, our approach preserving local volumes is more adequate than that for the global volume. 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 parallel computation is applicable.

5 Lattice-based 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.

5.1 Dynamic movement and lattice shape matching

A lattice of cubic cells containing the surface mesh is presented in the previous section. Now, we further define shape matching region for each particle. Each particle p is associated with a shape matching region $Region_p$. $Region_p$ contains p and all particles within a Manhattan distance \hat{w} from particle p . For instance, when $\hat{w} = 1$, $Region_p$ is the adjacent neighbors.

The main lattice shape matching algorithm is proposed by Rivers and James [23]. At each time step, each $Region$ 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:

$$g_p = \frac{1}{|Region_p|} \left(\sum_{r \in Region_p} \tilde{T}_r \right) x_p^0 \tag{13}$$

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 (14) and (15)

$$v_p(t+h) = v_p(t) + \alpha \frac{g_p(t) - x_p(t)}{h^2} + h \frac{f_p(t)}{m_p} \tag{14}$$

$$x_p(t+h) = x_p(t) + h v_p(t+h) \tag{15}$$

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

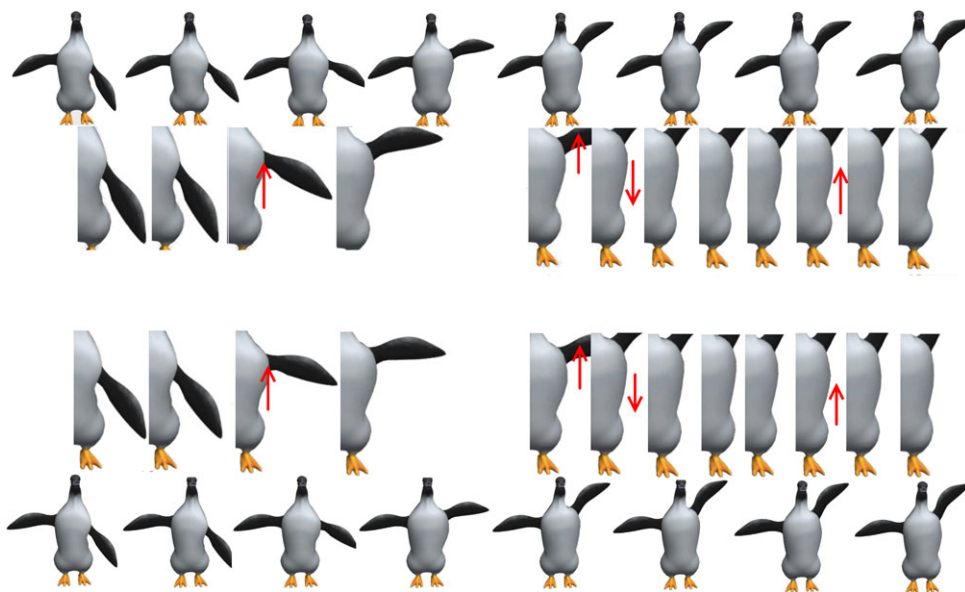
Applying the calculation of the dynamics to all particles results in a “gummy bear” like deformation. To embed the skeleton into the cells, we further assign the particles within bone cylinders to be rigidly adhered on the bone. The effect of “bone rigidity” for other particles depends on the region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

5.2 Combination of skinning and dynamic movement

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

$$\hat{x}_p = \delta x_p + (1 - \delta) \bar{x}_p \tag{16}$$

Fig. 10 Result of combination of skinning and lattice shape matching. *Upper:* $\delta = 0.7$; *lower:* $\delta = 0.9$. Check the muscle located near the wings. With larger δ , the secondary deformation performs more observably



where \bar{x}_p , the primary deformation, is the position evaluated in Sect. 4, and δ is the ratio of secondary deformation and $0 \leq \delta \leq 1$. If $\delta = 0$, the result is the same as for 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.

At each time step, each particle p vibrates between x_p and \bar{x}_p . The goal position g_p will be more and more close to \bar{x}_p . x_p will gradually converge toward \bar{x}_p . Finally, we compute the vertex positions of the mesh by barycentric coordinates. Besides, we also include the damping force described in [23] to speed up the convergence. Figure 10 shows an example. Left to right shows a motion sequence of a penguin waving its wing. In two different δ settings, we can observe that when the lower penguin has larger δ , its muscles vibrate larger, as the secondary deformation performs observably.

5.3 GPU-acceleration

For the sake of rendering performance, we use a vertex buffer object and a trilinear lattice deformer based on GLSL. The interpolation weight of vertices are considered as vertex attributes and could be stored at GPU memory during pre-processing stage. The positions of all particles are stored in a texture and are updated at each frame.

6 Experiment and result

Our approach is flexible since we provide an interactive environment and various adjustable mesh parameters with defaults for users. Figure 3 shows our subject mesh, skeleton, and mesh-skeleton mapping. The skeleton motion data

we used are from CMU's motion capture database [31], and a 30 Hz dataset for interactive applications mentioned in [16]. Besides, the posture of skeleton can also be adjusted by forward kinematics. Please browse our demo video, where we show skinning, combination of primary and secondary movement, and exaggerated deformation on characters' chest and arms. The influence of secondary deformation is adjustable as shown in the penguin movement.

Figure 11 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 to be about 1900 particles to get balance between skinning quality and performance.

Figure 12 shows the results of volume correction. The volume preservation method is capable of alleviating the joint collapsing effect resulting from linear blend skinning. Table 1 shows performance tests for various voxelization resolutions on a laptop and desktop.

7 Conclusion and future work

In this paper, we present a lattice-based framework for efficient surface deformation in skeleton-driven character animation. After voxelizing an input surface mesh with cells, our system automatically generates lattice-based skinning weights through diffusion-based influence propagation. To reduce the over-compressed artifacts, a hierarchical method is employed for volume preservation on deformable parts. The skinning deformation is then combined with dynamic particles for lattice shape matching to approximate the physically realistic secondary deformation.

The proposed system requires the triangle mesh and skeleton as input, and it can generate appealing surface

Fig. 11 Skinning artifacts. *Left:* with lower voxel resolution (1072 particles); *right:* with higher voxel resolution (1906 particles)



Fig. 12 The deformation result. *Left:* without hierarchical volume preservation; *right:* with hierarchical volume preservation. The volumes near the joints are closer to the original ones

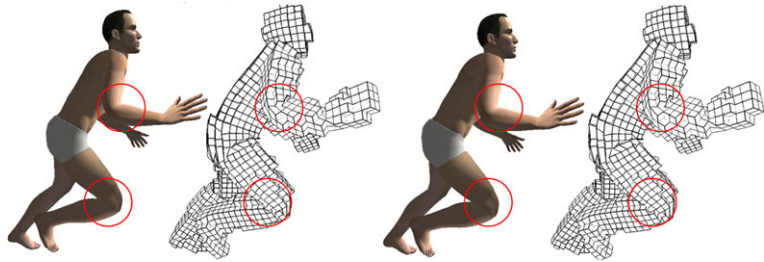






Table 1 Performance test on a moderate laptop and a desktop

	# 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: 16 ms. Number of joints: 38; Region size: 2 (cell width)			CPU: Intel Core 2 Duo P8600 RAM:DDR3-1066 4 GB RAM:DDR2-800 8 GB	CPU: Intel Core 2 Quad Q6600 Display: Nvidia GeForce G105M Display: Nvidia GeForce 8800GT

deformation according to character motion data or user-specified postures. The experiment shows that our framework makes our system adequate for real-time computation even on a moderate laptop computer.

Comparing to existing methods, such as [33], we do not constrain the exact volume but approximate the skinning, and volume of deformation parts through cells. However, we provide a unified framework combining primary

deformation, secondary deformation and volume preservation. It is a plausible approximation and can avoid extra switching cost among different structure types. We consider that our approach is balance in efficiency and visual performance. On the other hand, the process stages in our framework can also be replaced by more precise evaluation algorithms if appropriate structure transition is applied.

A possible future work is adaptive voxelization. For instance, octree-based approaches can achieve irregular sampling. Reducing the resolution of interior particles could further improve the performance and keep the appearance quality as well. Besides, the delicate design of cells and smooth-transit of rigidity will benefit the result. Another work is related to skinning weights. Currently, our automatic weight computation supports only positive weights; more variety of deformation may be generated if negative weights are considered.

References

- Baran, I., Popović, J.: Automatic rigging and animation of 3d characters. *ACM Trans. Graph.* **26**(3), 72 (2007)
- Botsch, M., Pauly, M., Wicke, M., Gross, M.: Adaptive space deformations based on rigid cells. *Comput. Graph. Forum* **26**(3), 339–347 (2007)
- Capell, S., Green, S., Curless, B., Duchamp, T., Popović, Z.: Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph.* **21**(3), 586–593 (2002)
- Cordier, F., Magnenat-Thalmann, N.: A data-driven approach for real-time clothes simulation. In: *Proc. Pacific Graphics*, pp. 257–266 (2004)
- Faloutsos, P., van de Panne, M., Terzopoulos, D.: Dynamic free-form deformations for animation synthesis. *IEEE Trans. Vis. Comput. Graph.* **3**(3), 201–214 (1997)
- Forstmann, S., Ohya, J., Krohn-Grimberghe, A., McDougall, R.: Deformation styles for spline-based skeletal animation. In: *Proc. ACM SIGGRAPH/Eurographics Symp. on Computer Animation*, pp. 141–150 (2007)
- James, D.L., Barbič, J., Twigg, C.D.: Squashing cubes: automating deformable model construction for graphics. In: *Proc. ACM SIGGRAPH 2004 Conference, Sketches & Applications* (2004)
- Joshi, P., Meyer, M., DeRose, T., Green, B., Sanocki, T.: Harmonic coordinates for character articulation. *ACM Trans. Graph.* **26**(3), 71 (2007)
- Ju, T., Schaefer, S., Warren, J.: Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.* **24**(3), 561–566 (2005)
- Ju, T., Zhou, Q., van de Panne, M., Cohen-Or, D., Neumann, U.: Reusable skinning templates using cage-based deformations. *ACM Trans. Graph.* **27**(5), 122 (2008)
- Kavan, L., Collins, S., Žára, J., O’Sullivan, C.: Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph.* **27**(4), 105 (2008)
- Kim, B.-U., Feng, W.-W., Yu, Y.: Real-time data driven deformation with affine bones. *Vis. Comput.* **26**(6–8), 487–495 (2010)
- Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. *ACM Trans. Graph.* **21**(3), 473–482 (2002)
- Lewis, J.P., Corder, M., Fong, N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In: *Proc. ACM SIGGRAPH’00*, pp. 165–172 (2000)
- Li, J., Lu, G., Ye, J.: Automatic skinning and animation of skeletal models. *Vis. Comput.* **27**(6–8), 585–594 (2011)
- Lin, I.-C., Peng, J.-Y., Lin, C.-C., Tsai, M.-H.: Adaptive motion data representation with repeated motion analysis. *IEEE Trans. Vis. Comput. Graph.* **17**(4), 527–538 (2011)
- Magnenat-Thalmann, N., Laperrière, R., Thalmann, D.: Joint-dependent local deformations for hand animation and object grasping. In: *Proc. Graphics Interface’88*, pp. 26–33 (1988)
- Müller, M., Dorsey, J., Mcmillan, L., Jagnow, R., Cutler, B.: Stable real-time deformations. In: *Proc. ACM SIGGRAPH Symp. on Computer Animation*, pp. 49–54 (2005)
- Müller, M., Gross, M.: Interactive virtual materials. In: *Proc. Graphics Interface’04*, pp. 239–246 (2004)
- Molino, N., Bao, Z., Fedkiw, R.: A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph.* **23**(3), 385–392 (2004)
- O’Brien, J.F., Zordan, V.B., Hodgins, J.K.: Combining active and passive simulations for secondary motion. *IEEE Comput. Graph. Appl.* **20**(4), 86–96 (2000)
- Peng, J.-Y., Lin, I.-C., Chao, J.-S., Chen, Y.-J., Juang, G.-H.: Interactive and flexible motion transition. *Comput. Animat. Virtual Worlds* **18**(4–5), 549–558 (2007)
- Rivers, A.R., James, D.L.: Fastlsm: fast lattice shape matching for robust realtime deformation. *ACM Trans. Graph.* **26**(3), 82 (2007)
- Shi, X., Zhou, K., Tong, Y., Desbrun, M., Bao, H., Guo, B.: Example-based dynamic skinning in real time. *ACM Trans. Graph.* **27**(3), 29 (2008)
- Sifakis, E., Neverov, I., Fedkiw, R.: Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph.* **24**(3), 417–425 (2005)
- Takamatsu, K., Kanai, T.: Volume-preserving lsm deformations. In: *Proc. ACM SIGGRAPH ASIA’09 Sketches* (2009), article: 15
- von Funck, W., Theisel, H., Seidel, H.-P.: Elastic secondary deformations by vector field integration. In: *Proc. Eurographics Symp. on Geometry Processing’07*, pp. 99–108 (2007)
- Wang, R.Y., Pulli, K., Popović, J.: Real-time enveloping with rotational regression. *ACM Trans. Graph.* **26**(3), 55 (2007)
- Wang, Y.-S., Lee, T.-Y.: Curve-skeleton extraction using iterative least squares optimization. *IEEE Trans. Vis. Comput. Graph.* **14**(4), 926–936 (2008)
- Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., Shum, H.: Large mesh deformation using the volumetric graph laplacian. *ACM Trans. Graph.* **24**(3), 496–503 (2005)
- CMU GraphicsLab: Motion Capture Database. <http://mocap.cs.cmu.edu> (2011). Accessed, Dec. 2011
- Lee, J., Kim, M.-S., Yoon, S.-H.: Patches: character skinning with local deformation layer. *Comput. Animat. Virtual Worlds* **20**(2–3), 321–331 (2009)
- Hyun, D.-E., Yoon, S.-H., Chang, J.-W., Seong, J.-K., Kim, M.-S., Jüttler, B.: Sweep-based human deformation. *Vis. Comput.* **21**(8–10), 542–550 (2005)



Cheng-Hao Chen received the B.S. degree from National Sun Yat-Sen University. In 2010, he received the M.S. degree in computer science from National Chiao Tung University, Taiwan. His research interests include character motion, geometric deformation and 3D game programming.



Ming-Han Tsai received the B.S. and M.S. degree in computer science from National Chiao Tung University, Taiwan, in 2007 and 2009, respectively. He is now a Ph.D. student in the Department of Computer Science, National Chiao Tung University, Taiwan. His research interests include image-based modeling, image synthesis, and motion capture.



Pin-Hua Lu received the B.S. and M.S. degree in computer science from National Chiao Tung University, Taiwan, in 2009 and 2011, respectively. His research interests include character animation, image and video synthesis.



I-Chen Lin received the B.S. and Ph.D. degree in computer science from National Taiwan University, Taiwan, in 1998 and 2003, respectively. In 2005, he joined Department of Computer Science, National Chiao Tung University, Taiwan. He is currently an assistant professor. His research interests include computer graphics, animation, image-based modeling and virtual reality. He is a member of ACM SIGGRAPH and IEEE.