Pacific Graphics 2015
N. J. Mitra, J. Stam, and K. Xu
(Guest Editors)

Volume 34 (*2015*), *Number 7*

Multi-layer Lattice Model for Real-Time Dynamic Character Deformation

Naoya Iwamoto[†1,2], Hubert P. H. Shum[‡3], Longzhi Yang[§3] and Shigeo Morishima[¶2,4]

[1]Waseda University, Japan
[2]JST CREST, Japan
[3]Northumbria University, United Kingdom
[4]Waseda Research Institute for Science and Engineering, Japan

Abstract

Due to the recent advancement of computer graphics hardware and software algorithms, deformable characters have become more and more popular in real-time applications such as computer games. While there are mature techniques to generate primary deformation from skeletal movement, simulating realistic and stable secondary deformation such as jiggling of fats remains challenging. On one hand, traditional volumetric approaches such as the finite element method require higher computational cost and are infeasible for limited hardware such as game consoles. On the other hand, while shape matching based simulations can produce plausible deformation in real-time, they suffer from a stiffness problem in which particles either show unrealistic deformation due to high gains, or cannot catch up with the body movement. In this paper, we propose a unified multi-layer lattice model to simulate the primary and secondary deformation of skeleton-driven characters. The core idea is to voxelize the input character mesh into multiple anatomical layers including the bone, muscle, fat and skin. Primary deformation is applied on the bone voxels with lattice-based skinning. The movement of these voxels is propagated to other voxel layers using lattice shape matching simulation, creating a natural secondary deformation. Our multi-layer lattice framework can produce simulation quality comparable to those from other volumetric approaches with a significantly smaller computational cost. It is best to be applied in real-time applications such as console games or interactive animation creation.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

1. Introduction

Simulating the movement of deformable characters has become a popular research topic in the past decade due to the increasing demand from the computer animation and games industries. With the advancement of computer graphics hardware, it becomes possible to simulate high quality character movement with realistic dynamic deformation.

Early research focused on simulating deformation based on the primary movement of the characters, which is the active movement produced by body parts. A popular approach known as *skinning* is used to map surface vertices onto the underlying skeleton structure. Given a skeletal movement, the deformation of the skin can be analytically calculated. However, skinning cannot generate involuntary secondary deformation such as jiggling of fat resulted from the primary movement. Considering the walking movement of a fat character as an example, without secondary deformation, the tummy would hang around the pelvis solidly as if it is made of hard muscle. With the raising expectation from the industry, secondary deformation has become an important research topic.

Simulating secondary deformation is a challenging problem. On one hand, while traditional volumetric simulation methods such as the finite element model can generate accurate dynamic deformation, the simulation stability depends on small time steps and therefore relatively higher computational cost. On the other hand, while *shape matching* based methods such as lattice shape matching can produce plausi-

† iwamoto@toki.waseda.jp
‡ hubert.shum@northumbria.ac.uk (corresponding author)
§ longzhi.yang@northumbria.ac.uk
¶ shigeo@waseda.jp

ble deformation in real-time, they are originally designed to simulate passive deformation due to external forces and collisions. Recent attempts to utilize shape matching algorithms for active character simulation produce unsatisfying results. The major problem is that if the particles representing the character are too stiff, simulation artifacts occur due to the use of high gains in a dynamics system. Oppositely, if they are too soft, the particles cannot catch up with the primary movement, resulting in cloth-like muscles. In general, it is not feasible to find a single stiffness value that can produce different range and speed of movement.

In this paper, we propose a new multi-layer lattice model to simulate high quality primary and secondary deformation in real-time. Our system first voxelizes the input character mesh into multiple anatomical layers, including the bone, muscle, fat and skin. It then performs primary deformation on the voxels belonging to the bone layer using *lattice-based skinning*. The movement of the bone voxels is propagated to the other layers under *lattice shape matching* simulation, generating realistic secondary deformation. Here, we use different stiffness and damping parameters for the muscle, fat and skin layer to generate the appropriate deformation behaviour. We also design a set of position-based constraints to ensure simulation quality. Since our multi-layer framework can better approximate the dynamics of real-life human body, it creates better quality secondary deformation.

Experimental results demonstrate that our framework can simulate realistic character movement from high dynamic captured motion, as shown in Figure 1. Since lattice shape matching is unconditionally stable, our framework can be applied for noisy motion captured from the Microsoft Kinect during run-time. Comparing to existing researches focusing on real-time simulation, our method produce deformation simulations of significantly higher quality thanks to the mutli-layer model.

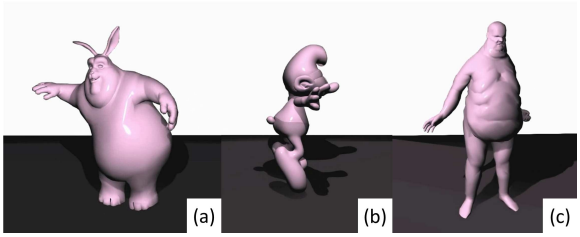

Figure 1: *Dynamic character deformation simulated by our method with captured motion: (a) bunny (b) Smurf (c) bloat.*

1.1. Contributions

In this paper, we present two major contributions:

- We propose a new real-time volumetric framework for dynamic character deformation, in which we produce primary and secondary deformation with different subsets of voxels. Voxels of simulated secondary deformation are driven by those of analytically calculated primary deformation, resulting in a robust dynamic system.
- We propose a new multi-layer model for lattice shape matching simulation, which allows us to adjust the simulation parameters for each anatomical layer independently. The model better approximates real-life body structure, and therefore produces higher quality secondary deformation.

2. Related work

In this section, we review works that are related to this research. We first discuss about dynamics simulation for passive objects. Since character movement requires both active and passive movement, we explain the algorithms for primary and secondary deformation respectively. Finally, we review multi-layer systems in the scope of character deformation, and point out how they can improve simulation quality.

2.1. Position-based Dynamics Simulation

Traditional dynamics simulation is force-based, in which the position of an object is calculated by internal and external forces using Newton's second law of motion [NMKC05]. A major problem is that in computer graphics, many applications require direct manipulation of positions. *Position-based dynamics* is proposed to avoid overshooting and energy gain problems due to the time-integration of force-based systems, and thereby improving the simulation stability [MHHR07]. With position-based dynamics, it becomes feasible to simulate deformable objects by representing them with volume-based particles known as voxels [MMCK14]. Different constraints are implemented to simulate complex dynamic systems involving strain [MCKM14] and bending/twisting of elastic rods [USS14]. In this paper, we develop position-based constraints to produce stable and plausible simulation.

Oriented particles apply position-based dynamics to particles with respective orientations [MC11b]. The major advantages of oriented particles over traditional voxel-based simulation is that realistic simulation can be performed using only the particles sampled on the surface of the object, thereby enhancing computation speed. This approach is particular suitable for simulating thin layer such as hair and clothes [MC11a]. However, if the object is consist of multiple layers of materials like human body, multiple layers of particles are required. This deflects the purpose and the advantage of using only a small number of particles. To relieve the problem, displacement functions are proposed to indicate the attenuation of deformation resulted from the oriented particles, describing how skeleton deforms muscle [FGBP11]. Still, setting up such functions for a multi-layer system is complex and non-trivial. In this work, we

explicitly classify voxels into different layers, enabling an intuitive design process and a simple simulation framework with improved performance.

Shape matching approaches simulate deformable object by moving the voxels towards their respective predicted positions in the next time step using positional-based constraints [MHTG05]. Extending this idea, lattice shape matching is proposed [RJ07, SOG08]. The idea is to group voxels called lattice into overlapping blocks. When the position of one voxel is changed, the movement affects the whole block, which subsequently affects the overlapping neighbour blocks. Rigidity of an object can thus be adjusted by determining the block size and hence the level of overlapping. In this paper, we propose a customize lattice shape matching algorithm that considers multiple voxel layers of an actively moving character.

2.2. Dynamic Character Deformation

The fundamental difficulty of simulating dynamic character deformation is that there are both active and passive components in the movement. Active movement such as a walking motion is known as primary movement, while uncontrolled passive movement such as jiggling of fat is known as secondary movement. Here, we review related researches in the scope of skeleton driven character movement.

The process to simulate primary movement is commonly known as skinning. In mesh-based approach, the positions of the surface vertices are expressed as a weighted sum of skeleton effectors with a set of automatically or semi-automatically calculated weight [JBK*12]. To enhance the visual quality of the deformation, energy functions are designed to better estimate the skinning weight [KS12]. For character mesh with complex topology, it is proposed to voxelize the mesh and represent the skinning weight with bone to skin distance calculated by Dijkstra's algorithm [DdL13]. We adapt [DdL13] to perform primary deformation as it is compatible with other voxel-based deformation algorithms. However, we only apply primary deformation on the innermost layer of voxels in our multi-layer framework.

Secondary movement requires dynamics simulation as it is driven by physical constraints. Rig-space is proposed as a subspace for deformations, and finite element discretization is applied to calculate the dynamic deformation [HMT*12, HTC*13]. With some manually created samples of physical behaviour, simulate quality can be further enhanced [SZT*08]. The major disadvantage of these methods is that surface mesh instead of voxels is used to approximate the volume of the character. It is difficult to design multiple layers of meshes such as muscle and fat, as well as to simulate the interaction between them.

As a solution, volumtric methods are proposed. Skeleton driven musculoskeletal system can be simulated with body-centered cubic tetrahedral lattice [TSB*05] or Eulerian Tissue [FLP14], which represents the muscle geometry. By considering the interaction between the skin geometry and the environment, it is possible to alter the primary movement when there is an external force of impact [LYWG13]. Since many computer graphics application are driven by positional constraints, position-based dynamics is used to create stable and robust simulations [RF14]. Voxel representation can be applied to improve solver performance and memory efficiency in soft tissue deformation [MZS*11].

The common bottleneck of finite element method based approaches is the computational complexity. Methods such as [TSB*05, FLP14, LYWG13, MZS*11] are too computational costly for limited hardware system such as game console in real-time applications. Recent advancement in the field allows real-time performance using reduced deformable body models, linear-time skeleton dynamics and explicit integration [KP11]. However, as a force-based framework, the size of the time step affects the simulation stability, and small time steps dramatically increase the computational cost. We prefer shape matching based approaches that are unconditionally stable for any size of time step. Another advantage of our proposed framework is that it requires only a few easy-to-tune parameters, and hence local minima can easily be avoided when optimizing the parameters.

Chen et al. conduct secondary deformation using lattice shape matching [CTLL13]. While the system is fast and stable, the create results show minimal secondary movement. Similarly, in the infant model with rigid skeleton experiment of [RJ07], secondary movement cannot be effectively produced, and the character is rigid unless the user applies external forces. These are because of the rigidity design of the lattice shape matching framework. High rigidity is needed in order to drive character movement from the skeleton, which creates minimal secondary movement. If the rigidity is lowered, the lattices cannot catch up with the skeleton and result in cloth like muscle. In this paper, we propose a multi-layer framework with different rigidity for each layer to solve the problem.

2.3. Multi-layer System

The idea of using multi-layer system in character animation has been proposed to enhance the quality of primary movement. By representing the major muscles of the human body as an inner layer, the outer layer skin deformation is more realistic as anatomic deformation involving muscle movement is included [PCLS05]. By analyzing the anatomy structure of a character, it becomes possible to transfer the underlying skeletal structure from one character to another, thereby achieving automatic skinning [AHLG*13]. Accurate anatomical volume model can be obatined by 3D MRI scans, such that volume deformation can be simulated with the actual skeletal structure [RLNN11].

Multi-layer algorithms can also be used to simulate secondary movement. By introducing automatically generated

inner layers into a mesh object, collisions handling and volume preservation can be improved [DB13]. In order to simulate realistic muscle behaviour, it is proposed to model muscle with Non Uniform Rational B-Spline (NURBS) surface generator, and apply spring equations to simulate muscle jiggling and skin tension [MH07]. Combining simulation with manually created blend shape, muscle simulation speed can be enhanced [MH07]. In [AS07], manually created anatomic tissue layer known as vortex deformers is proposed. The layer can model the dynamics of incompressible anatomic material, and is designed alongside with the skeleton hierarchy. Despite of the simulation quality enhancement in these research, it is inefficient to define volumetric layers under a mesh-based framework. Furthermore, the simulation quality depends heavily on the mesh topology and structure. We propose an efficient multi-layer system under a volumetric framework, and apply lattice shape matching to simulate secondary movement in real-time.

3. Method Overview

The overview of our system is shown in Figure 2. The system takes the mesh of a character and a skeletal motion as the input. Unlike traditional voxelization processes, our system voxelize the mesh into multiple anatomical layers (Section 4). The bone layer is used for primary deformation with respect to the input motion (Section 5). The rest of the layers are fed into the secondary deformation system using different simulation parameters (Section 6). During secondary deformation, we first apply lattice shape matching with the result from primary deformation, such that the voxels from the bone layer drive the movement of those from other layers (Section 6.1). Then, we design a set of position-based constraints to ensure the quality of the simulated results (Section 6.2).

4. Multi-Layer Voxelization

There are multiple anatomical tissue layers in human beings and animals, which behave differently during active movement. Given the same motion sequences, two characters with the same appearances but different proportion of tissue layer should deform differently. For instance, a character with more fat produces more soft body effects such as jiggling. This observation leads us to a multiple layer lattice model that enhances simulation quality by treating each tissue layer differently based on their physical features.

We first voxelize the mesh into a set of voxels, and then classify them into different anatomical layers. Notice that while we define four layers in this work, the proposed framework can handle any number of layers depending on the animator's needs, as long as the innermost bone layer is available.

We combine the use of both conservative voxelization and solid voxelization [SS10] in order to generate a voxelized model that can produce stable simulation. We first apply conservative voxelization that includes all voxels overlapping or touching the input mesh. We then apply solid voxelization to include extra voxels for constructing a watertight voxel model. This allows us to create an inclusive (i.e. by including voxels either overlap or touch the character mesh) and watertight voxel model of the character. Figure 3 shows some examples of voxelized characters used in this paper.

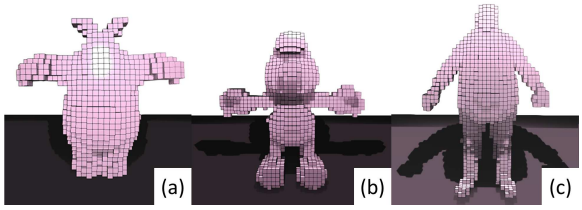

Figure 3: *Voxelization of (a) bunny (b) Smurf (c) bloat.*

Once the voxel model of a character is generated, we classify the voxels into multiple layers using a semi-automatic procedure with the following three steps: (1) Taking the rest pose and the corresponding skeletal structure as the input, the bone layer is defined as the set of voxels that are located within a user-specified distance from the skeletal structure. Here, we use Manhattan distance as it considers the voxel connectivity structure. (2) The skin layer is defined as a watertight layer at the surface of the voxelized model with the thickness of one voxel. This layer is useful to maintain surface tension, thereby preserving the fine surface details such as facial elements of a character during simulation. (3) For the rest of the voxel, we classify them into the muscle and fat layer based on a user-defined ratio, with voxels closer to the bone belonging to the muscle, and those closer to the skin belonging to the fat. By adjusting the ratio of these two layers, we can generate different deformation behaviours. Figure 4 shows the result of voxel classification for the bunny character.

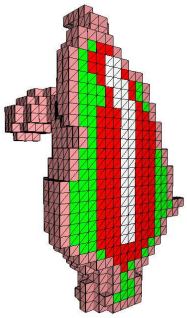

Figure 4: *Our proposed multi-layer structure with bone (white), muscle (red), fat (green) and skin (pink) layers.*

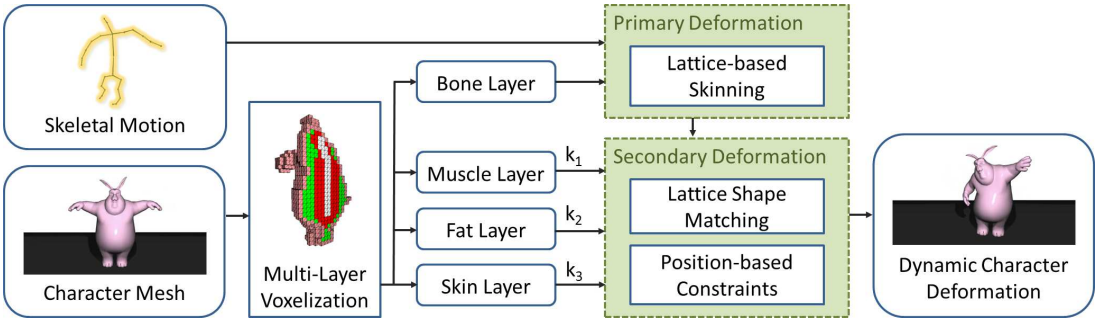

Figure 2: *The overview of our real-time dynamic character deformation framework.*

In our system, the animator provides the parameters required for layer classification, including the bone width and the muscle to fat ratio. The animator can define such parameters in a body part level, such that the different part of the character can behave differently during the simulation. For example, the stomach of the bunny character should contain more fat than its arm.

5. Primary Deformation

We implement primary deformation with lattice-based skinning, which is a skinning method based on lattice voxels rather than traditional triangle mesh. We follow [DdL13] as it can efficiently estimate skinning weight using voxel connectivity. Here, we outline the algorithm in [DdL13] and point out the differences in our implementation.

5.1. Skinning Weight Computation

The first step of primary deformation is to calculate the skinning weight of the voxels, which describes in what extend the movement of the skeletal structure affects the voxels. Unlike [DdL13], primary deformation is only applied in the bone layer in this work.

We take the voxelized model in a rest-pose and its corresponding skeletal structure as the input. Given a bone from the skeletal structure b and a bone layer voxel i. We define d_{b-i} as the Manhattan distance between voxel i and the nearest voxel that touches the bone b, which is calculated by Dijkstra's algorithm. Since we only apply primary deformation in the innermost layer of the character body, we do not need to evaluate the distance between a surface voxel to the character's mesh skin as in [DdL13].

The influence weight from the bone b to the voxel i is calculated as:

$$w_{b-i} = \left[\frac{1}{(1-\alpha)(d_{b-i}{}') + \alpha(d_{b-i}{}')^2}\right]^2 \quad (1)$$

$$d_{b-i}{}' = d_{b-i} + \varepsilon \quad (2)$$

where ε is a small value to avoid zero being used as denominator, $\alpha \in [0,1]$ is a hand-tuned constant to adjust the attenuation of the distance influence. It is set as 0.5 for all experiments in our system.

We prefer Manhattan distance [DdL13] to Euclidean distance [CTLL13]. This is because two geometrically near voxels are not necessarily near through tissue connection. Euclidean-based skinning weight is therefore not accurate in areas of complex structure, such as the underarms of a character.

5.2. Lattice-based Skinning

The second step of primary deformation is to apply the calculated skin weight for repositioning voxels based on the movement of the skeletal structure. Because of the use of Lattice-based Manhattan distance, this process is known as Lattice-based skinning.

The deformed position of a voxel i is calculated as a weighted sum of bone transformations:

$$\mathbf{p}_i = \left(\sum_{b=1}^{b_{total}} w_{b-i} T_b\right)\mathbf{p}_i^0 \quad (3)$$

where b_{total} is the total number of bone, T_b is the transformation matrix of the bone b, $\mathbf{p}_i^0$ is the position of the voxel in the rest pose, w_{b-i} is calculated in Equation 1. We normalize the weight w_{b-i} such that the sum of weight influenced from the bones is 1.0.

An example of the skinning result is shown in Figure 5. Notice that since the process is only applied for bone layer voxels, the process is efficient. Also, while it is possible to enforce straight bones by customizing the weight function, we deliberately allow bendable bones using traditional lattice-based skinning. This is because under our multi-layer lattice framework, the bones are used to drive the movement of outer layer voxels with secondary deformation. If straight bones are used, the acute angles formed between bones would decrease the simulation quality of the voxels around. We found that bendable bones ease the problem and produce better results.

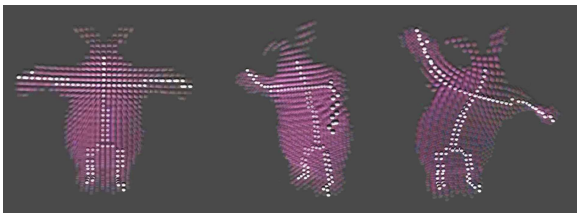

Figure 5: *Visualization of lattice-based skinning applied on the bone layer only.*

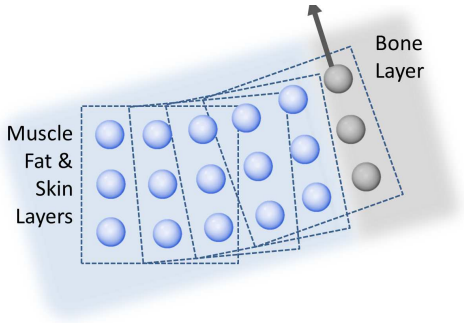

Figure 6: *The bone voxels driving the movement of other voxels under lattice shape matching.*

6. Secondary Deformation

In this section, we explain how we simulate the secondary deformation of the character using position-based dynamics [MHHR07] that is unconditionally stable. This involves a customized lattice shape matching algorithm to drive the muscle, fat and skin based on the movement of the bone voxels, as well as a set of positional-based constraints to ensure simulation quality.

Algorithm 1 Position-updating procedure

1: **for each** voxel i in muscle, fat and skin layers **do**
2: $\hat{\mathbf{p}}_i^{t+h} \leftarrow \mathbf{p}_i^t + h\mathbf{v}_i^t$ // Predict the next position
3: **end for**
4: **for** n:1 **to** n_{total} **do**
5: SatisfyStretchConstraints($\hat{\mathbf{p}}^{t+h}$)
6: SatisfyVolumeConstraints($\hat{\mathbf{p}}^{t+h}$)
7: LatticeShapeMatching($\hat{\mathbf{p}}^{t+h}$)
8: **end for**
9: **for each** votex i in muscle, fat and skin layers **do**
10: $\mathbf{p}_i^{t+h} \leftarrow \hat{\mathbf{p}}_i^{t+h}$ // Update position
11: $\mathbf{v}_i^{t+h} \leftarrow \frac{\mathbf{p}_i^{t+h} - \mathbf{p}_i^t}{h}$ // Update velocity
12: $\mathbf{v}_i^{t+h} \leftarrow k_d \mathbf{v}_i^{t+h}$ // Damp velocity
13: **end for**

The simulation framework considers the muscle, fat and skin layers and involves three major parts as shown in Algorithm 1. The first part (line 1 to 3) obtains the predicted voxel positions in the next time step $\hat{\mathbf{p}}_i^{t+h}$ based on the current position $\mathbf{p}_i^t$, velocity $\mathbf{v}_i^t$ and the time-step h. The second part (line 4 to 8) is to repeatedly apply lattice shape matching and constraints, such that the predicted voxel positions can satisfy all required criteria. The final part (line 9 to 13) is to calculate the position $\mathbf{p}_i^{t+h}$ and velocity $\mathbf{v}_i^{t+h}$ of the next time-step, which involves damping the velocity using a damping parameter k_d. In our system, k_d is hand-tuned for each simulation layer.

While traditional systems such as [MHHR07] solve the lattice shape matching and constraints multiple times (n_{total} in line 4), we find that solving the system once (i.e. $n_{total} = 1$) creates results of reasonable quality with minimal computational cost. A useful strategy here is to place the lattice shape matching after the constraints such that the movement generated is not overridden, while the constraints are to be maintained in the next iteration.

In the following, we explain the process of lattice shape matching and constraints satisfaction in more details.

6.1. Lattice Shape Matching

Here, we explain our customized lattice shape matching (LSM) model to drive voxels to their target position based on the movement of the bone voxels. LSM is used as it is stable and computationally efficient, making it suitable for real-time simulation. Our LSM method is based on [RJ07], but it is designed to work under our multi-layer lattice model. Here, we consider each voxel as one lattice in the LSM model.

The idea of LSM is to group voxels into a number of overlapping lattice regions, and apply shape matching to estimate the overall region movement whenever the position of its voxels is changed. Since the regions are overlapping, the movement of one region propagates to other regions. Figure 6 shows an example, in which we group every 3×3 voxels into a lattice region, resulting in four overlapping regions. The movement of the grey voxel lead to the movement of its corresponding region, and the positions of the rest of the voxels in the region are affected. Since the regions are overlapping, the position change propagates to all the blue voxels.

Considering a voxel i belonging to a lattice region R_i with a manually set region width d_r, the goal position of voxel i is calculated as:

$$g_i = \frac{1}{|R_i|}\left(\sum_{r \in R_i} T_r\right)\mathbf{p}_i^0, \tag{4}$$

where $|R_i|$ indicates the number of voxels inside the region, T_r represents the transformation of voxel r in R_i, $\mathbf{p}_i^0$ is the rest position of voxel i. In most cases, $|R_i| = d_r \times d_r \times d_r$, except for regions that are close to the characters surface.

With the goal position calculated, we define the position correction function as:

$$\Delta \mathbf{p}_i = k_s (\mathbf{g}_i - \mathbf{p}_i) \tag{5}$$

where k_s is the stiffness parameter. In our system, k_s is hand-tuned for each simulation layer. We calculate the goal positions and correct the positions of all simulating voxels based on their corresponding stiffness.

Comparing with the original work [RJ07], our customized approach has two major differences to fit into the multi-layer framework. First, the voxels of the bone layer are included in the LSM model, but their positions are not updated. Instead, their positions are calculated directly from the primary deformation as explained in Section 5. The movement of bone voxels is then propagated to other layers because of the overlapping lattice regions under the LSM model. Second, voxels of different simulation layers (including muscle, fat and skin) have different stiffness parameters, k_s. This allows us to control the deformation behaviour of different layers.

6.2. Constraints

In this section, we explain the stretch constraint and the volume preservation constraint in our system.

Before going into the details of the constraint definition, we explain the constrain solver we adapt from [MHHR07]. The solver allows us to calculate the position update $\Delta \mathbf{p}_i$ of a voxel i in order to satisfy a constraint, which is therefore known as a correction function:

$$\Delta \mathbf{p}_i = -w_i \frac{C(\mathbf{p})}{\sum_j w_j |\nabla_{\mathbf{p}_j} C(\mathbf{p})|^2} \nabla_{\mathbf{p}_i} C(\mathbf{p}) \tag{6}$$

in which the three inputs of the equation are: (1) the constraint equation $C(\mathbf{p})$, (2) its derivative $\nabla_{\mathbf{p}_i} C(\mathbf{p})$, and (3) the weight of the voxel w. The input of the constraint and its derivative can be the position of one or more voxels, depending on the constraint definition.

6.2.1. Stretch Constraint

The stretch constraint is implemented as maintaining the distance between a voxel with its neighbours. In other words, it is implemented as a set of distance constraints.

In previous works, stretch constraints have been implemented by considering the 6 adjacent neighbours that share a face with the voxel [DBB11]. However, we found that such an implementation leads to under-constraint voxels in thin body parts as there are not enough voxels with shared faces, such as the ears of a bunny character and the arms of a Smurf character. To solve the problem, we consider a set of 20 neighbours that share either a face or a corner with the voxel.

We follow [MHHR07] to implement the distance constraint. The constraint between the positions of two voxels $\mathbf{p}_i$ and $\mathbf{p}_j$ is defined as:

$$C(\mathbf{p}_i, \mathbf{p}_j) = |\mathbf{p}_i - \mathbf{p}_j| - d^0_{i-j} \tag{7}$$

where d^0_{i-j} is a constant value representing the length between the voxels in the rest pose. The derivatives of the constraint are:

$$\begin{aligned} \nabla_{\mathbf{p}_i} C(\mathbf{p}_i, \mathbf{p}_j) &= \frac{\mathbf{p}_i - \mathbf{p}_j}{|\mathbf{p}_i - \mathbf{p}_j|} \\ \nabla_{\mathbf{p}_j} C(\mathbf{p}_i, \mathbf{p}_j) &= -\frac{\mathbf{p}_i - \mathbf{p}_j}{|\mathbf{p}_i - \mathbf{p}_j|} \end{aligned} \tag{8}$$

The weight w is set as the inverse of voxel mass, which is 1.0 in our system. Therefore, $w_i = w_j = 1$.

Substituting the constraint, the derivative of the constraint and the weight to Equation 6 gives us the correction function:

$$\begin{aligned} \Delta \mathbf{p}_i &= -\frac{w_i}{w_i + w_j} (|\mathbf{p}_i - \mathbf{p}_j| - d^0_{i-j}) \frac{\mathbf{p}_i - \mathbf{p}_j}{|\mathbf{p}_i - \mathbf{p}_j|} \\ \Delta \mathbf{p}_j &= +\frac{w_i}{w_i + w_j} (|\mathbf{p}_i - \mathbf{p}_j| - d^0_{i-j}) \frac{\mathbf{p}_i - \mathbf{p}_j}{|\mathbf{p}_i - \mathbf{p}_j|} \end{aligned} \tag{9}$$

For each voxel, we apply Equation 8 on its 20 neighbours. The final position correction is the mean value calculated from all neighbours.

6.2.2. Volume Preservation Constraint

In order to preserve the volume of the character, we design the volume preservation constraint to maintain the volume of the voxels. While Rumman et al. have proposed a volume constraint based on the tetrahedral structure [RF14], it cannot be applied to voxel based simulation directly as voxels have a cubic structure. Here, we design a new voxel-based constraint to implement volume preservation under the framework proposed in [RF14].

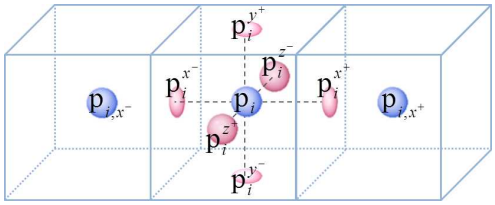

Figure 7: *Volume constraint of a voxel.*

For each voxel i, we calculate six voxel surface points $\mathbf{p}_i^{x^+}$, $\mathbf{p}_i^{x^-}$, $\mathbf{p}_i^{y^+}$, $\mathbf{p}_i^{y^-}$, $\mathbf{p}_i^{z^+}$, $\mathbf{p}_i^{z^-}$ by evaluating the mean distance between $\mathbf{p}_i$ and the positions of the six adjacent voxels, as shown in Figure 7.

The volume constraint of a voxel $\mathbf{p}_i$ is calculated as its current volume subtracted by its initial volume:

$$C(\mathbf{p}_i) = V(\mathbf{p}_i) - v_i^0 \tag{10}$$

where v_i^0 is the volume of the voxel i in the rest pose of the

character, $V(\mathbf{p}_i)$ is the volume of the voxel i at the current frame, which is calculated using the surface points [TK11]:

$$\begin{aligned} V(\mathbf{p}_i) = &\mathbf{p}_i^{x^+} \cdot \left(\mathbf{p}_i^{y^+} \times \mathbf{p}_i^{z^+}\right) + \mathbf{p}_i^{x^+} \cdot \left(\mathbf{p}_i^{y^-} \times \mathbf{p}_i^{z^-}\right) \\ &+ \mathbf{p}_i^{x^+} \cdot \left(\mathbf{p}_i^{y^+} \times \mathbf{p}_i^{z^-}\right) + \mathbf{p}_i^{x^-} \cdot \left(\mathbf{p}_i^{y^-} \times \mathbf{p}_i^{z^+}\right) \\ &+ \mathbf{p}_i^{z^+} \cdot \left(\mathbf{p}_i^{y^+} \times \mathbf{p}_i^{x^-}\right) + \mathbf{p}_i^{z^+} \cdot \left(\mathbf{p}_i^{y^-} \times \mathbf{p}_i^{x^+}\right) \\ &+ \mathbf{p}_i^{z^-} \cdot \left(\mathbf{p}_i^{y^+} \times \mathbf{p}_i^{x^+}\right) + \mathbf{p}_i^{z^-} \cdot \left(\mathbf{p}_i^{y^-} \times \mathbf{p}_i^{x^-}\right) \end{aligned} \quad (11)$$

The derivatives of the volume constraint in all directions are calculated as:

$$\begin{aligned} \nabla_{\mathbf{p}_i^{z^-}} C(\mathbf{p}_i) &= -\mathbf{p}_i^{x^+} \times \mathbf{p}_i^{y^-} - \mathbf{p}_i^{x^-} \times \mathbf{p}_i^{y^+} + \mathbf{p}_i^{y^+} \times \mathbf{p}_i^{x^+} + \mathbf{p}_i^{y^-} \times \mathbf{p}_i^{x^-} \\ \nabla_{\mathbf{p}_i^{z^+}} C(\mathbf{p}_i) &= -\mathbf{p}_i^{x^+} \times \mathbf{p}_i^{y^+} - \mathbf{p}_i^{x^-} \times \mathbf{p}_i^{y^-} + \mathbf{p}_i^{y^+} \times \mathbf{p}_i^{x^-} + \mathbf{p}_i^{y^-} \times \mathbf{p}_i^{x^+} \\ \nabla_{\mathbf{p}_i^{y^-}} C(\mathbf{p}_i) &= \mathbf{p}_i^{x^+} \times \mathbf{p}_i^{z^-} + \mathbf{p}_i^{x^-} \times \mathbf{p}_i^{z^+} + \mathbf{p}_i^{z^+} \times \mathbf{p}_i^{x^+} + \mathbf{p}_i^{z^-} \times \mathbf{p}_i^{x^-} \\ \nabla_{\mathbf{p}_i^{y^+}} C(\mathbf{p}_i) &= \mathbf{p}_i^{x^+} \times \mathbf{p}_i^{z^+} + \mathbf{p}_i^{x^-} \times \mathbf{p}_i^{z^-} + \mathbf{p}_i^{z^+} \times \mathbf{p}_i^{x^-} + \mathbf{p}_i^{z^-} \times \mathbf{p}_i^{x^+} \\ \nabla_{\mathbf{p}_i^{x^-}} C(\mathbf{p}_i) &= \mathbf{p}_i^{y^+} \times \mathbf{p}_i^{z^-} + \mathbf{p}_i^{y^-} \times \mathbf{p}_i^{z^+} - \mathbf{p}_i^{z^+} \times \mathbf{p}_i^{y^+} - \mathbf{p}_i^{z^-} \times \mathbf{p}_i^{y^-} \\ \nabla_{\mathbf{p}_i^{x^+}} C(\mathbf{p}_i) &= -\nabla_{\mathbf{p}_i^{z^-}} C(\mathbf{p}_i) - \nabla_{\mathbf{p}_i^{z^+}} C(\mathbf{p}_i) - \nabla_{\mathbf{p}_i^{y^-}} C(\mathbf{p}_i) \\ &\quad - \nabla_{\mathbf{p}_i^{y^+}} C(\mathbf{p}_i) - \nabla_{\mathbf{p}_i^{x^-}} C(\mathbf{p}_i) \end{aligned} \quad (12)$$

The weight w_i, which is also known as the lattice volume correction scale, is calculated as

$$w_i = 1 - \frac{d_{i-skin}}{d_{max-skin}} \quad (13)$$

where d_{i-skin} is the Manhattan distance from voxel i to the closest skin voxel, $d_{max-skin}$ is the longest distance from any voxel to the closest skin voxel. The weight is therefore larger for voxels closer to the skin, allowing us to maintain surface shapes better.

We finally substitute the constraint, its derivatives, as well as the weight to Equation 6 to generate the correction function. This is performed in the software implementation and the results follow.

7. Experimental Results

In this section, we evaluate our system and compare the simulation quality with other systems. All experiments were conducted using a desktop computer with a 3.4 GHz Intel Core i7-3770 CPU, 8GB RAM and a GeForce GTX 560 Ti graphic card. The character meshes we used were obtained from the Blender foundation and the website http://tf3dm.com/.

7.1. Character Simulation

We demonstrate our system with three character models including the Smurf, the bunny and the bloat, as shown in Figure 1. The character meshes are voxelized based on a pre-defined resolution that is fine enough to represent the mesh details. For each simulation, the voxelization resolution, number of voxels, lattice region width d_r (as explained in Section 6.1) and the frame time in *ms* excluding mesh drawing are shown in Table 1. Except the Smurf that requires a higher voxel resolution to capture the thin arms, our system runs faster than real-time (33*ms* per frame). The average ratio of the computational cost for each process is shown in Figure 8.

Table 1: *Simulation Details*

Model	Res.	Voxels	d_r	*ms*
Bunny	32	3,671	3	19.47
Smurf	64	4,312	3	36.36
Bloat	32	4,798	3	32.45
Bloat (Low Res.)	16	922	2	5.47

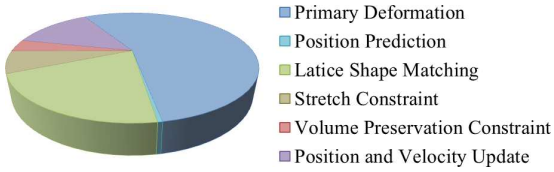

Figure 8: *The computational cost of individual process.*

We also show that it is easy to balance the trade-off between computational speed and simulation quality by adjusting the voxelization resolution of the bloat as shown in the last two rows of Table 1. Figure 9 shows the simulated result. Notice that if the value is too low, the voxels cannot represent the fine details of the mesh, and therefore deformation quality is reduced. Extremely low resolution should be avoided, as the system may group different body parts into the same voxel.

Table 2: *Stiffness and Damping Parameters*

Model	k_s^{mus}	k_s^{fat}	k_s^{skin}	k_d^{mus}	k_d^{fat}	k_d^{skin}
Bunny	1.8	0.5	1.7	0.6	0.5	0.4
Smurf	3.2	1.2	2.8	0.4	0.5	0.7
Bloat	3.3	0.7	3.0	0.7	0.5	0.6
Bloat (Low Res.)	1.9	0.9	1.3	0.6	0.4	0.3

Table 2 shows the stiffness k_s in Equation 5 and the damping k_d in Algorithm 1 we used for the muscle, fat and skin layers. Tuning the values for each layer is efficient, as there exists a wide range of usable values. Our principle for k_s is that $k_s^{mus} > k_s^{skin} >> k_s^{fat}$, and k_d is similar for all layers. Then, we adjust individual values to obtain a movement behaviour that is suitable for the character, like the bunny hav-

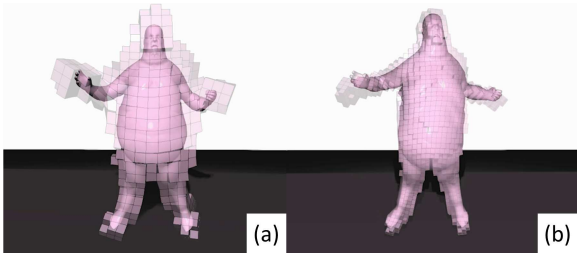

Figure 9: *Comparison between (a) low resolution simulation, and (b) normal simulation.*

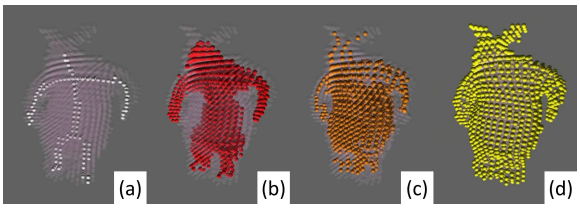

Figure 10: *Visualization of voxels belonging to different layers, including (a) bone, (b) muscle, (c) fat, (d) skin.*

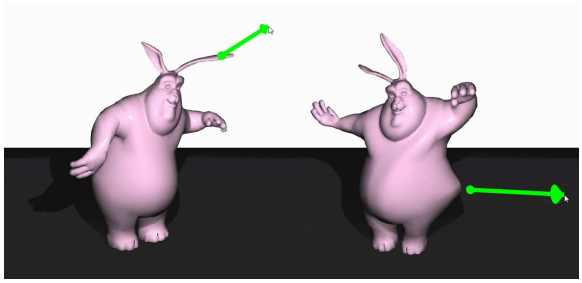

Figure 11: *Run-time user interaction by mouse dragging.*

Figure 12: *Real-time simulation using motion from Kinect.*

ing a softer body than the Smurf in general. Higher voxel resolution tends to require larger k_s and k_d, such that we can maintain rigidity across more voxels.

We visualize the voxels allocated for different layers of the bunny character in Figure 10. Notice that the arms are mostly made of muscle voxels while the tummy has more fat voxels. As a result, during the simulation, the arms are more rigid while the tummy is softer.

We provide a user interface to interact with the simulation by pulling the character with a small amount of force, as shown in Figure 11. We also allow the user to control the character movement using the motion captured by Kinect in real-time, as shown in Figure 12. Even with the noisy, unfiltered input captured by Kinect, we can produce high quality real-time deformation thanks to the robust simulation.

7.2. Comparisons

Here, we compare our system with other papers and setups to evaluate its performance.

We compare our system with a baseline system with no secondary deformation [DdL13], as shown in Figure 13. As expected, due to the lack of dynamic simulations, [DdL13] cannot simulate jiggling of soft body parts such as the ears and the tummy of the bunny character.

We also compare with [CTLL13] as shown in Figure 14. Due to the lack of multi-layer system, we use the stiffness of muscle for all voxels. As shown in Figure 14a, since [CTLL13] does not have a stretch constraint, the voxels cannot catch up with the body movement. We increase the voxels gains such that they can keep up with the movement as shown in Figure 14c. However, due to the use of large gains in a dynamics system, the system becomes unstable and voxels are overshot.

In general, without the mutli-layer framework, we found that it is difficult to find a stiffness value that can produce high quality simulation for all the dancing movements. This is because without the anatomical information of the body structure, the simulation engine cannot approximate real-life dynamics accurately.

The skin layer is important in maintaining surface tension. As shown in Figure 15, with only the bone, muscle and fat layers, the surface of the character is over-deformed during high dynamic movement, resulting in distorted facial elements. The surface tension provided by the skin layer also helps maintaining shapes of thin body parts such as the ears of the bunny character.

8. Conclusion and Discussions

In this paper, we present a new real-time framework for simulating dynamic character deformation. We introduce a novel multi-layer voxel model, and apply position-based dynamics to create high quality and robust results. In our system, we apply primary deformation to the voxels belonging to the bone layer. During secondary deformation, the movement of the bone layer is propagated to the muscle, fat and skin layers using lattice shape matching and a set of position-

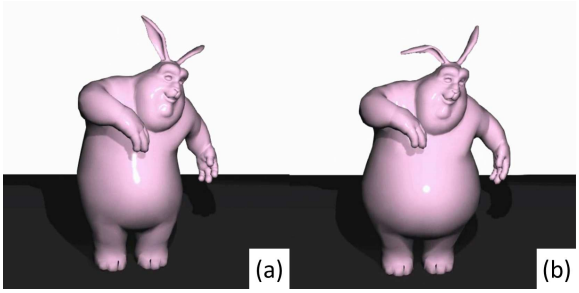

Figure 13: *Comparison between (a) [DdL13] and (b) our method.*

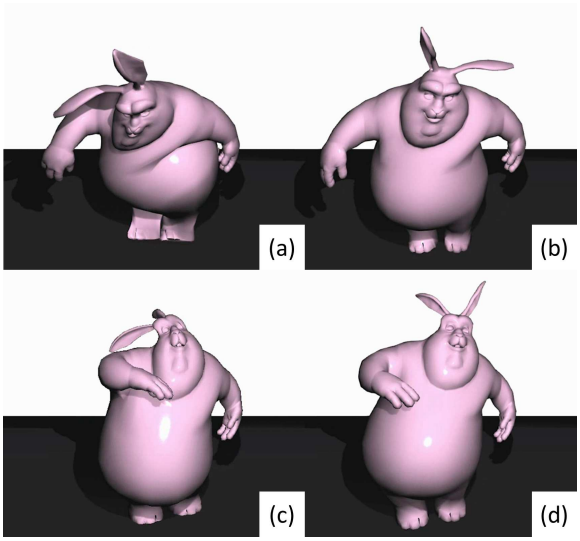

Figure 14: *Comparison between (a) [CTLL13] with normal stiffness and (b) our method, as well as between (c) [CTLL13] with higher stiffness and (d) our method.*

based constraints. We demonstrate higher quality character deformation comparing to existing real-time algorithms in the field.

We approximate the anatomical structure of a character using separated tissue layers. However, for real-life animals, the muscle and fat are not separable. It is possible to enhance our system by introducing the voxels that are made of both muscle and fat, in which the stiffness and damping are interpolated based on the muscle-fat proportion. This, however, will increase the character design complexity, and we opt for a simpler representation.

Currently, the thickness of different anatomical layers is either procedurally or manually defined. It is possible to obtain more realistic information such as the bone width by analyzing real-life animals with magnetic resonance imaging (MRI) as shown in [RLNN11, FLP14]. One potential application is to generate virtual characters by obtaining the anatomical information of real humans. The body deformation will therefore be realistic, and it can serve as an alternative method for performance animation.

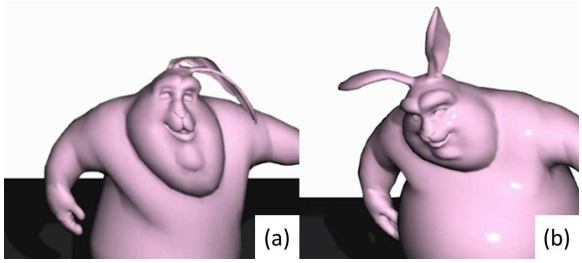

Figure 15: *Comparison between (a) our method without the skin layer and (b) our method.*

As in real-life, our system drives the character body with bone movement. Therefore, a more realistic bone structure can enhance the simulation quality. In fact, the human body has a rib cage as oppose to the simplified backbone structure commonly used in computer animation. Including such a kind of structure can improve the accuracy of the body movement, and is one of our future research direction.

Tuning the voxel resolution requires the user to trade-off computational speed and deformation quality. In the future, we would like to explore the use of adaptive volume representation [SOG08] in our multi-layer framework, such that body parts with finer details such as fingers can be represented with higher voxel resolution. One potential challenge is the complexity of tuning simulation parameters such as stiffness and damping under the adaptive multi-layer framework.

We would also like to further enhance the computational speed by implementing GPU-based solutions in the future. Voxel operations can easily be parallelized and therefore are highly compatible with GPU computation.

Acknowledgement

This project was supported in part by the Engineering and Physical Sciences Research Council (EPSRC) Ref: EP/M002632/1, and in part by CREST, JST.

References

[AHLG*13] ALI-HAMADI D., LIU T., GILLES B., KAVAN L., FAURE F., PALOMBI O., CANI M.-P.: Anatomy transfer. *ACM Trans. Graph. 32*, 6 (nov 2013), 188:1–188:8. 3

[AS07] ANGELIDIS A., SINGH K.: Kinodynamic skinning using volume-preserving deformations. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 129–140. 4

[CTLL13] CHEN C.-H., TSAI M.-H., LIN I.-C., LU P.-H.: Skeleton-driven surface deformation through lattices for real-time character animation. *The Visual Computer 29*, 4 (2013), 241–251. 3, 5, 9, 10

[DB13] DEUL C., BENDER J.: Physically-based character skinning. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (Lille, France, nov 2013), Eurographics Association. 4

[DBB11] DIZIOL R., BENDER J., BAYER D.: Robust real-time deformation of incompressible surface meshes. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, ACM, pp. 237–246. 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. 3, 5, 9, 10

[FGBP11] FAURE F., GILLES B., BOUSQUET G., PAI D. K.: Sparse meshless models of complex deformable solids. *ACM Trans. Graph. 30*, 4 (July 2011), 73:1–73:10. 2

[FLP14] FAN Y., LITVEN J., PAI D. K.: Active volumetric musculoskeletal systems. *ACM Trans. Graph. 33*, 4 (July 2014), 152:1–152:9. 3, 10

[HMT*12] HAHN F., MARTIN S., THOMASZEWSKI B., SUMNER R., COROS S., GROSS M.: Rig-space physics. *ACM Trans. Graph. 31*, 4 (jul 2012), 72:1–72:8. 3

[HTC*13] HAHN F., THOMASZEWSKI B., COROS S., SUMNER R. W., GROSS M.: Efficient simulation of secondary motion in rig-space. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2013), SCA '13, ACM, pp. 165–171. 3

[JBK*12] JACOBSON A., BARAN I., KAVAN L., POPOVIĆ J., SORKINE O.: Fast automatic skinning transformations. *ACM Trans. Graph. 31*, 4 (jul 2012), 77:1–77:10. 3

[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

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

[MC11a] MÜLLER M., CHENTANEZ N.: Adding physics to animated characters with oriented particles. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (2011), pp. 83–91. 2

[MC11b] MÜLLER M., CHENTANEZ N.: Solid simulation with oriented particles. *ACM Trans. Graph. 30*, 4 (jul 2011), 92:1–92:10. 2

[MCKM14] MULLER M., CHENTANEZ N., KIM T.-Y., MACKLIN M.: Strain Based Dynamics. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation* (2014), Koltun V., Sifakis E., (Eds.), The Eurographics Association. 2

[MH07] MILLER E., HARKINS J.: Musculo-skeletal shape skinning. In *ACM SIGGRAPH 2007 Sketches* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. 4

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (apr 2007), 109–118. 2, 6, 7

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (jul 2005), 471–478. 3

[MMCK14] MACKLIN M., MÜLLER M., CHENTANEZ N., KIM T.-Y.: Unified particle physics for real-time applications. *ACM Trans. Graph. 33*, 4 (jul 2014), 153:1–153:12. 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 (jul 2011), 37:1–37:12. 3

[NMKC05] NEALEN A., MÜLLER M., KEISER R., CARLSON E. B. M.: Physically based deformable models in computer graphics. In *Eurographics 2005. STAR - State of the Art Reports* (2005), pp. 71–94. 2

[PCLS05] PRATSCHER M., COLEMAN P., LASZLO J., SINGH K.: Outside-In Anatomy Based Character Rigging. In *Symposium on Computer Animation* (2005), Terzopoulos D., Zordan V., Anjyo K., Faloutsos P., (Eds.), The Eurographics Association. 3

[RF14] RUMMAN N. A., FRATARCANGELI M.: Position based skinning of skeleton-driven deformable characters. In *Proceedings of the 30th Spring Conference on Computer Graphics* (New York, NY, USA, 2014), SCCG '14, ACM, pp. 83–90. 3, 7

[RJ07] RIVERS A. R., JAMES D. L.: Fastlsm: Fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph. 26*, 3 (jul 2007). 3, 6, 7

[RLNN11] RHEE T., LEWIS J. P., NEUMANN U., NAYAK K.: Scan-based volume animation driven by locally adaptive articulated registrations. *IEEE Transactions on Visualization and Computer Graphics 17*, 3 (May 2011), 368–379. 3, 10

[SOG08] STEINEMANN D., OTADUY M. A., GROSS M.: Fast adaptive shape matching deformations. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2008), SCA '08, Eurographics Association, pp. 87–94. 3, 10

[SS10] SCHWARZ M., SEIDEL H.-P.: Fast parallel surface and solid voxelization on gpus. In *ACM SIGGRAPH Asia 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH ASIA '10, ACM, pp. 179:1–179:10. 4

[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

[TK11] TAKAMATSU K., KANAI T.: A method for volume-preserving deformations by shape matching. *The Journal of the Institute of Image Electronics Engineers of Japan 40*, 4 (2011), 549–557. 8

[TSB*05] TERAN J., SIFAKIS E., BLEMKER S. S., NG-THOW-HING V., LAU C., FEDKIW R.: Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics 11*, 3 (May 2005), 317–328. 3

[USS14] UMETANI N., SCHMIDT R., STAM J.: Position-based elastic rods. In *ACM SIGGRAPH 2014 Talks* (New York, NY, USA, 2014), SIGGRAPH '14, ACM, pp. 47:1–47:1. 2