Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2013)
J. Bender, J. Dequidt, C. Duriez, and G. Zachmann (Editors)

Physically-Based Character Skinning

Crispin Deul and Jan Bender

Graduate School CE, TU Darmstadt, Germany

Abstract

In this paper we present a novel multi-layer model for physically-based character skinning. In contrast to geometric approaches which are commonly used in the field of character skinning, physically-based methods can simulate secondary motion effects. Furthermore, these methods can handle collisions and preserve the volume of the model without the need of an additional post-process. Physically-based approaches are computationally more expensive than geometric methods but they provide more realistic results. Recent works in this area use finite element simulations to model the elastic behavior of skin. These methods require the generation of a volumetric mesh for the skin shape in a pre-processing step. It is not easy for an artist to model the different elastic behaviors of muscles, fat and skin using a volumetric mesh since there is no clear assignment between volume elements and tissue types. For our novel multi-layer model the mesh generation is very simple and can be performed automatically. Furthermore, the model contains a layer for each kind of tissue. Therefore, the artist can easily control the elastic behavior by adjusting the stiffness parameters for muscles, fat and skin. We use shape matching with oriented particles and a fast summation technique to simulate the elastic behavior of our skin model and a position-based constraint enforcement to handle collisions, volume conservation and the coupling of the skeleton with the deformable model. Position-based methods have the advantage that they are fast, unconditionally stable, controllable and provide visually plausible results.

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

1. Introduction

Skinning is a fundamental process in the content creation pipeline of feature films, video games or in the special effects industry. In these application areas animated virtual humans play an important role. For an animation a skeleton of a virtual character and a skin that defines its external shape is required. The deformation of the skin is computed for every frame of the animation depending on the actual positions of the bones and joints of the skeleton. In this context skinning describes the mapping of the skeleton motion onto the skin. Since skinning has to be performed in each frame, fast mapping algorithms are essential.

In 1988 Magnenat-Thalman et al. [MTLT88] described one of the first skinning algorithms which is called linear blend skinning (LBS). Due to its simplicity and speed LBS is even used today. Though, LBS has also some problems like volume loss at articulated joints. Many following works like dual-quaternion skinning [KCvO08] improve the LBS algorithm. The approaches above are geometrically motivated and lack the visualization of phenomena like bulging of the flesh near colliding body parts or jiggling of fatty tissue.

Several extensions to LBS enhance the skinning output in a post-process. They handle self-collisions near articulated joints, apply volume preservation techniques, or compute secondary motion effects to improve the visual plausibility. However, the dependence on the LBS output may still lead to artifacts and the post-processing approaches can only address a few of the phenomena of real skin at a time.

In contrast to LBS, example-based methods blend deformed skin examples from a database depending on the skeletal pose [PH08]. The problem of this technique is to create the database that must cover the whole intended motion space of the skeleton.

A promising way to perform realistic skinning is to use a physically-based simulation. The simulation can compute the elastic behavior of skin, perform volume preservation, and handle collisions at the same time. Most recent works

DOI: 10.2312/PE.vriphys.vriphys13.025-034

rely on the finite element method (FEM) to simulate the skin deformation [MZS*11, KP11] at near-interactive frame rates. The main benefit of FEM is the promise of realistic deformations. However, the computational effort of FEM-methods is quite high.

This paper presents a novel skinning method using physically-based simulation. Our main goals are to perform skinning at interactive rates and to reduce the work for the animation artist as well as the implementation effort. Therefore, we introduce a multi-layer skin model to represent the skin and to approximate the interplay of fat, muscles, and bone tissue. As the layers are automatically derived from a rigged surface mesh and a corresponding animation skeleton, the artist does not need to create an additional simulation mesh. Consequently, the method fits nicely into the standard animation pipeline. In contrast to earlier approaches we use the concept of oriented particles in conjunction with a position-based dynamics solver to simulate the behavior of skin with respect to articulation of the animation skeleton. While the elastic behavior of skin, fat, and muscles is modeled by oriented particle deformations, the motion of the animation skeleton is transferred by LBS skinning onto the innermost layer. The oriented particle deformations are solved together with additional local volume preservation and contact constraints in order to obtain a plausible deformation behavior near articulated joints. The advantage of performing the physically-based simulation using position-based methods is that these methods are very fast, unconditionally stable and controllable [BMOT13]. Therefore, complex models can be simulated at interactive rates.

Our contributions:

- A multi-layer skin model that exploits the performance advantage of fast summation techniques to reach interactive performance.
- A framework to simulate phenomena of real skin like bulging or jiggling by solving the interplay of collisions, local volume preservation, and skin dynamics.

2. Related Work

Geometric Skinning Starting with the work of Magnenat-Thalmann et al. [MTLT88], linear blend skinning (LBS) has been widely used to model the visual representation of animated characters. LBS is easy to implement and has low computational requirements but suffers under severe visual artifacts like collapsing of the skin at joints or the so called candy-wrapper effect at twisting bones. Different approaches [MMG06, KCvO08, JBK*12] have been presented to solve these problems. In a recent work Kavan and Sorkine [KS12] developed new joint-based deformers that describe non-linear spatial deformations at the joints. Together with an optimization of the skinning weights to approximate the behavior of elastic solids the method generates superior results compared to earlier work. However, skinning techniques can not handle deformations caused by collisions, preserve volume or capture dynamic effects.

Example-Based methods To solve these problems several example-based methods have been developed, e.g. [LCF00, KJP02, PH08]. These approaches derive the positions of the skin vertices not solely from the actual state of the animation skeleton. Instead, the vertex positions are interpolated based on samples in a database. Filling the database with samples is the drawback of most example-based methods. Either artists have to create the samples by hand for a wide variety of poses, simulate the motion in an offline process or expensive motion capture sessions with a dense marker set are required.

Post-Processing Post-processing approaches try to correct or enhance the results of LBS techniques in a second step. In contrast to example-based methods these approaches do not need additional input data like a database of examples.

Von Funck et al. [vFTS08] correct the volume loss near joints. They define a displacement field and move the vertices of the mesh along this field. Chen et al. [CLTL11] use the volume correction of [TK09] which is an extension of [vFTS08] for a lattice. Their goal is to generate secondary motion of the skin by using fast lattice shape-matching (FastLSM) [RJ07] for the simulation of the elastic behavior. The result of FastLSM is then blended with a lattice-based skinning technique. Finally, the volume is corrected in a last step.

The approaches above do not handle self-collisions. In cases of strong articulation interpenetrations may occur at joints and cause visual artifacts. Vaillant et al. [VBG*13] propose an approach based on the projection of particles onto isosurfaces of volumetric implicit representations of the body. Besides the prevention of interpenetrations the method can also produce bulges next to contact areas. This approach shows convincing results in preventing the volume loss of LBS and generating contact regions. However, in some situations LBS can generate a skin configuration where the projection fails.

While all of the post-processing methods show real-time performance and convincing results, they only address some of the issues of LBS techniques at the same time. Furthermore, they directly depend on the results of LBS which may lead to degraded results of the algorithm.

Physically-Based Skinning Volume conservation and contact handling have been active research topics in the area of physically-based simulation for many years [BFA02, ISF07, DBB09, TMOT12]. Since physically-based methods also provide the possibility of secondary motion effects, these methods are investigated in the field of skinning as well. However, the simulation of a whole dynamics system leads to much more computational effort than basic LBS skinning, example-based skinning, or LBS with post-processing.

Nevertheless, several works reach almost interactive performance.

Capell et al. [CBC*07] use a linear FEM control lattice that aligns with the bones to simulate the deformation of skin. Their method also supports contact handling but lacks the preservation of volume. Moreover, the construction of the control lattice is a laborious task for the artist. Kim and Pollard [KP11] use nonlinear FEM in their work to obtain a high modeling accuracy of biological tissue. They use a symplectic Euler integrator to circumvent problems in deriving the stiffness matrix of their FEM model. A drawback of this integrator is the requirement of small timesteps to get a stable simulation with complex models. In order to circumvent the drawback of small time steps McAdams et al. [MZS*11] use implicit time stepping that is stable even with large time steps. Furthermore, they prefer a uniform hexahedral lattice over a tetrahedral mesh to facilitate the parallelization of their code on modern hardware. Additionally, to improve the performance and stability of the simulation, they introduce a one-point quadrature scheme and a multigrid solver. Despite all of this effort their method works at best at near interactive performance. Nevertheless, they show very convincing deformations of skin.

The idea to use a layered model to simulate articulated characters is not new. In the Critter system [CHP89] the points of a control lattice for free-form deformation of the skin are connected by springs with each other and the animation skeleton. A more sophisticated model that uses a mass spring lattice with three layers together with volume preserving constraints is presented by Terzopoulos and Waters [TW91] to generate facial animations. Turner and Thalmann [TT93] model the elasticity of skin with the finite difference technique and simulate the fat layer by Hookian spring forces. However, they treat muscles as purely controlled elements. Thus, they do not model muscles with deformable methods. Galoppo et al. [GOT*07] define a two layer model of skin and bones to perform a global collision response of the skin and skeleton. However, the expression of strain in pose space does not capture bulging at flexed joints.

In contrast to previous work, we use position-based methods to model the elasticity of deformable tissue. These methods are unconditionally stable. As a result, we do not suffer under overshooting problems of force-based methods and need no reduced time step size when using a fast explicit time integrator. Furthermore, we disburden the artist by automatically computing a suitable simulation model. Our skinning method supports secondary motion effects, contact handling and volume preservation.

3. Multi-Layer Skin Model

We introduce a multi-layer skin model to approximate the diverse physical properties of the several tissue layers found in the human body. Our model results from a trade-off between performance and reduced artistic work on the one hand and the accuracy in simulating biological tissue on the other hand. We want to capture the behavior of this tissue to some extent to compute plausible skin deformations near articulated body parts. At most parts of the human body four types of tissue, namely skin, fat, muscles and bones, define the outer appearance of a person. To simulate the deformable types of tissue most recent approaches that use FEM require the generation of tetrahedral or hexahedral meshes. Instead of imposing the burden of creating a volumetric mesh and defining different kinds of tissue for the volume elements on the animation artist, we directly use the output of the standard animation pipeline. The input for our deformable model is a given closed triangular mesh of a virtual character and an according animation skeleton. Based on the mesh three additional layers are created (see Figure 1). For each vertex of the original mesh the nearest point on the center line of the corresponding bone is computed. This bone is chosen depending on the skinning weights of the mesh vertices. To automatically compute these weights we use the approach of Baran and Popović [BP07]. The original vertex and the point on the center line define a projection segment for the creation of new layers. Along the projection segment copies of the original mesh vertices are created for the additional layers. On each new layer the projected vertices are connected according to the topology of the original mesh. The resulting four layers represent from the outside to the inside: skin, fat, muscular tissue, and the interface between muscle and bone. The thickness of the three inner layers can be adjusted through a projection parameter that defines the relative position of each new vertex on the projection segment. This allows an artist to model the torso or a fat belly, for example. In case of the torso, the artist would chose small projection parameters. As a result, there would be thin fat and muscle layers and the innermost bone layer, in close proximity to the skin, would represent the chest/ribcage sufficiently.

We simulate the elastic behavior of each layer by region-based shape matching with oriented particles. This approach uses overlapping regions of particles to compute the deformations. On each layer of our model we create a particle for each vertex in the corresponding mesh and define a region for this particle. This region contains the ω-ring of the particle defined by the mesh of the layer, where ω is the region size. Furthermore, to couple the layers we include sibling particles from adjacent layers into the region (see Figure 2). These particles lie on the same projection segment.

The derivation of tissue layers from a triangle mesh allows us to apply fast summation techniques to speed up the dynamics computations. Fast summation methods have previously been used in the skinning approach of Chen et al. [CLTL11]. Their approach applies the fast summation on a uniform grid. However, to capture fine detail the grid has to be refined globally. As a result, the computational effort increases significantly with every refinement. In contrast, our

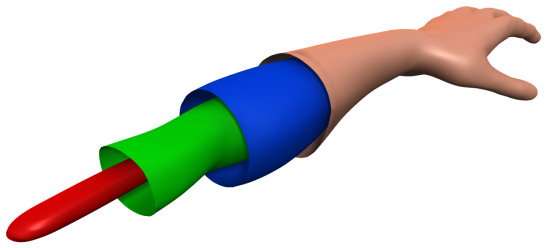


Figure 1: *Multi-layer skin model of an arm: The three internal mesh layers (blue, green, and red) are automatically generated by copying the outer skin layer. Then, the vertices of each generated mesh are projected closer to the associated bone of the rigged skeleton. The staggered cut view shows the input skin mesh as well as the generated layers which represent fat (blue), muscles (green), and the interface between bone and muscles (red). At the vertices of this innermost layer the kinematic movement of the rigged skeleton is coupled to the deformable model.*

model supports local refinement without a drastic impact on the performance.

The mass of a particle in our model is determined by approximating the volume that the particle represents in the model. First, the area A represented by a particle in its corresponding layer is determined by using Voronoi regions and a special treatment for obtuse triangles [MDSB03]. Then, the half distance to each coupled layer is summed up to get the height h. Finally, we approximate the volume by $V = A \cdot h$ and compute the mass by $m = V \cdot \rho$, where ρ is the density.

4. Dynamic Simulation

For the simulation of the dynamic behavior of our skin model we combine two position-based methods: shape matching with oriented particles [MC11b] and position-based constraint enforcement [MHHR07]. Oriented particles are used to simulate the elastic behavior of skin. We use distance constraints to realize the coupling between the skin model and the animated skeleton as well as to model the bones near joints. Unilateral contact constraints and volume constraints are defined to perform collision handling and volume conservation. These constraints are enforced by position-based dynamics. Position-based methods are fast, unconditionally stable, controllable and provide visually plausible results. This makes them well-suited for interactive physically-based character skinning.

4.1. Overview

On each layer of our skin model we use a mesh of particles for the simulation. Each particle has a mass m, a position $\mathbf{x}$ and a velocity $\mathbf{v}$. A simulation step for the multi-layer model is performed as described by algorithm 1, where $\mathbf{a}$ are external accelerations acting on the particles and Δt is the time step size.

Algorithm 1 Simulation step

1: **for all** vertices i **do**
2: $\quad \mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \Delta t\, \mathbf{a}_i$
3: $\quad \mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \Delta t\, \mathbf{v}_i^{n+1}$
4: **end for**
5: collisionDetection($\mathbf{x}^{n+1}$)
6: **for** $k = 1$ **to** $N_{\text{elasticity}}$ **do**
7: $\quad$ shapeMatching($\mathbf{x}^{n+1}$)
8: $\quad$ projectDistanceConstraints($\mathbf{x}^{n+1}$)
9: $\quad$ projectContactConstraints($\mathbf{x}^{n+1}$)
10: **end for**
11: **for** $k = 1$ **to** N_{volume} **do**
12: $\quad$ projectVolumeConstraints($\mathbf{x}^{n+1}$)
13: **end for**
14: **for all** vertices i **do**
15: $\quad \mathbf{v}_i^{n+1} \leftarrow (\mathbf{x}_i^{n+1} - \mathbf{x}_i^n)/\Delta t$
16: **end for**
17: dampVelocities($\mathbf{v}^{n+1}$)
18: velocityUpdate($\mathbf{v}^{n+1}$)

In lines (1)-(4) the time integration is performed using the symplectic Euler method. We then use a discrete collision detection to define the contact constraints for the current simulation step. Shape matching and position-based constraint enforcement for the distance and contact constraints are performed in an iterative process with a fixed number of $N_{\text{elasticity}}$ iterations in lines (6)-(10). After that the volume constraints are handled in a second loop with N_{volume} iterations. The volume correction takes the collisions into account to prevent interpenetrations due to position changes. After all position corrections the velocities of all particles are updated in lines (14)-(16). The resulting velocities are damped in line (17) for more realistic results. In the last step we make the velocity field divergence free to avoid oscillations and modify the velocities of colliding vertices to account for friction and restitution.

4.2. Simulation Methods

This section describes the position-based methods used for the simulation of our multi-layer skin model.

4.2.1. Oriented Particles

In this work we use oriented particles to model the elastic behavior of the skin. The concept of oriented particles extends the original shape matching approach of Müller et al. [MHTG05] by incorporating an orientation for every particle of the simulation mesh. The basic idea of shape matching is to first match the undeformed reference configuration

$\mathbf{x}^0$ of a particle cloud with the current deformed configuration $\mathbf{x}$ by a rigid body transformation to obtain goal positions. Then, each particle is pulled towards its corresponding goal position to simulate the elastic behavior of the deformable model.

The transformation is determined by the centers of mass $\mathbf{x}^0_{cm}$ and $\mathbf{x}_{cm}$ of the reference and the deformed configuration and the rotational part $\mathbf{R}$ of the affine transformation:

$$\mathbf{A} = \sum_i m_i \left(\mathbf{x}_i - \mathbf{x}_{cm}\right) \left(\mathbf{x}_i^0 - \mathbf{x}_{cm}^0\right)^T, \tag{1}$$

which can be extracted by a polar decomposition. Finally, goal positions are computed for each particle i by

$$\mathbf{g}_i = \mathbf{T} \begin{bmatrix} \mathbf{x}_i^0 \\ 1 \end{bmatrix},$$

where $\mathbf{T} = \begin{bmatrix} \mathbf{R} & (\mathbf{x}_{cm} - \mathbf{R}\mathbf{x}_{cm}^0) \end{bmatrix}$. The particles are pulled towards their goal positions during the time integration by adding the term $s(\mathbf{g}_i - \mathbf{x}_i)/\Delta t$ to the equation in line (2) of algorithm 1, where $s \in [0,1]$ is a user-defined stiffness parameter.

If a shape matching step as described above is performed for all particles, only small deformations are possible. To simulate large deformations the original shape is subdivided in overlapping regions and goal positions are determined for each region. The final goal position of a particle is then obtained by blending the results of all regions which contain the particle.

If the particles in a region are nearly co-linear or co-planar, the matrix in equation (1) becomes ill-conditioned and its polar decomposition numerically unstable. The usage of oriented particles solves this problem [MC11b]. By adding the moment matrices $\mathbf{A}_i$ of the oriented particles in equation (1)

$$\mathbf{A} = \sum_i \left(\mathbf{A}_i + m_i \left(\mathbf{x}_i - \mathbf{x}_{cm}\right) \left(\mathbf{x}_i^0 - \mathbf{x}_{cm}^0\right)^T \right)$$

the polar decomposition becomes stable even for single particles.

The update of the additional orientation per particle leads to a slight increase in computational cost. However, an artist might model the different layers of our deformable model very flat near joints to represent the thickening of bones in this area. In this case, the usage of oriented particles is an essential extension, especially when the joints move and the skin stretches.

4.2.2. Position-Based Constraint Enforcement

Position-based dynamics enforces bilateral constraints $C(\mathbf{x}) = 0$ and unilateral constraints $C(\mathbf{x}) \geq 0$ by a direct modification of the particle positions in a model. If a constraint is not satisfied, the goal is to find a position correction $\Delta\mathbf{x}$ such that $C(\mathbf{x} + \Delta\mathbf{x}) = 0$. Nonlinear constraint functions are linearized by the following approximation $C(\mathbf{p} + \Delta\mathbf{p}) \approx C(\mathbf{p}) + \nabla_{\mathbf{p}} C(\mathbf{p}) \cdot \Delta\mathbf{p}$. Since the resulting system is under-determined, the direction of the position changes is restricted to the gradient $\nabla_{\mathbf{p}} C(\mathbf{p})$.

The position correction for particle i is determined by

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

where $w_i = 1/m_i$ is the inverse particle mass. For unilateral constraints we perform this position correction only if $C(\mathbf{x}) < 0$.

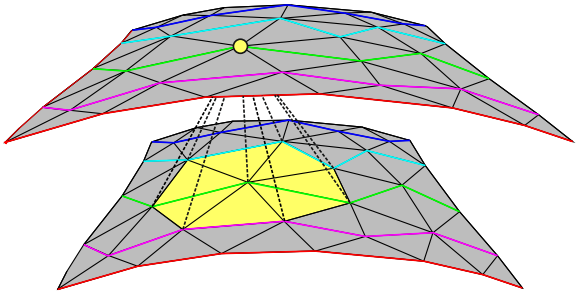


Figure 2: *Fast summation with our multi-layer model. To compute the sum for the yellow region, we first determine the prefix sums for the corresponding paths (cyan, green, purple). Then, for each path the difference of the border values is added to the sum of the region. Finally, the values of the vertices on the coupled layers (yellow sphere), which correspond to the center vertex of the region, are added.*

4.3. Fast Summation

When the size ω of the overlapping regions for shape matching increases, the computation of the sum for the center of mass $\mathbf{x}_{cm}$ and the sum in equation (1) becomes computationally expensive. The naive computation of these sums for all regions requires $O(\omega^d n)$ operations, where n is the number of regions and d the dimension of the mesh. For the fast summation, we exploit the fact that our multi-layer model consists of multiple triangle meshes ($d = 2$). Furthermore, we adapt the fast summation technique for triangle meshes of Diziol et al. [DBB11] which needs only $O(\omega n)$ operations. In a pre-processing step this method determines a path layout for the triangle mesh by using a heuristic such that each particle of the mesh is exactly part of one path (see Figure 2). Since we have the same topology on each layer, we have to determine the path layout only once. To perform the fast summation, we determine the prefix sum for each path. This can be done in parallel. If we subtract the value before a path enters a region from the value where it leaves the region, we get the sum of all values on the path in the region. Summing up the differences of all paths which intersect a region, we

get the total sum of a region in the triangle mesh. Since in our multi-layer model regions contain also particles of adjacent layers, we add the values of the corresponding particles in a final step (see Figure 2). The computation of the prefix sum as well as the total region sum is done in parallel.

4.4. Kinematic Coupling

Coupling constrains the movement of the skin layers relative to the bones of the animation skeleton. To couple our deformable body we determine goal positions for the innermost layer, the bone layer, by LBS. Throughout the iterations of the constraint enforcement loop the particle positions of the bone layer are moved to the LBS positions by distance constraints [Jak01] with a target distance of zero. Although these distance constraints support a stiffness parameter, we chose in our examples a value of one to enforce a strong coupling with the kinematic movement of the skeleton. Furthermore, we still perform shape matching of oriented particles for the bone layer even though the distance constraints pin the particles to the LBS positions. However, the oriented particle regions of the bone layer contain particles of the next layer which are affected by the shape matching computation. As a result, the shape matching step for the bone layer leads to a better coupling of the skeletal motion to the other layers of our skinning model.

The linear elasticity of our skinning model may lead to an unnatural loss of volume at the outside of bent joints. Here, the skin should be able to deform in tangential directions to the skin surface to represent stretching, but the distance to the bone should not change to much. However, the stiffness parameter of oriented particle regions works uniformly in all directions. In order to counteract the deflation, we attach the particles near joints with distance constraints to the joint center. Because we want to avoid the intersection of the tissue with the bones around the joint we define unilateral distance constraints. Accordingly, these constraints use a lower bound on the distance to the joint center. Consequently, particles may still move away from the joint and the tissue can deform into all but the joints direction.

4.5. Collision Handling

We use position-based constraints to handle collisions of our skin model. These constraints are solved together with the elastic constraints. Our own experiments with a collision resolution based on [BFA02] showed jittering artifacts in case of a self-colliding skin near joints. In contrast, the solution of collision constraints together with the elastic constraints leads to an equilibrium of constraint impulses, which solves the jittering problem.

In their work Müller and Chentanez [MC11b] propose a method to handle collisions of models simulated with oriented particles based on ellipsoids. Furthermore, they mention a bound of the aspect ratio of the ellipsoids of 2 : 1 from the largest principal radius to the smallest principal radius. These restrictions on the aspect ratio would force us to use large ellipsoids to cover the surface of our triangular mesh to miss no collision. In a recent work Müller and Chentanez [MC11a] present an improved algorithm for collision handling with surface meshes. Nevertheless, this algorithm still does not solve the restriction on the ellipsoid aspect ratio. Thus, it leads to visible gaps between self-colliding body parts at flexed joints of our model.

In this work we extend the collision resolution of [MHHR07]. This method defines a unilateral constraint for each point-triangle intersection. The constraints are resolved by a position-based solver. In our work we additionally introduce constraints to resolve collisions between two edges. These additional constraints allow us to obtain a completely intersection free state which is not guaranteed by only using point-triangle constraints. Furthermore, edge-edge-constraints improve the stability of our collision resolution. The constraint for the edge-edge collision is

$$C(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\mathbf{p}_4) = (\mathbf{p}_4-\mathbf{p}_1)\frac{\mathbf{n}}{|\mathbf{n}|},$$

where $\mathbf{p}_1$ and $\mathbf{p}_2$ are the vertices of the first edge and $\mathbf{p}_3$ and $\mathbf{p}_4$ are the vertices of the second edge. The vector $\mathbf{n} = (\mathbf{p}_2-\mathbf{p}_1)\times(\mathbf{p}_4-\mathbf{p}_3)$ is the normal of the plane defined by both edges. In the rare case of parallel edges, we use the perpendicular vector which intersects both edges as vector $\mathbf{n}$. For the points $\mathbf{p}_i$ we get the following gradients:

$$\begin{aligned}
\nabla_{\mathbf{p}_1} C(\mathbf{p}_1,\ldots,\mathbf{p}_4) &= \frac{1}{|\mathbf{n}|}\left(\mathbf{e}_{14}\mathbf{M}\mathbf{e}_{34}^*-\mathbf{n}\right)\\
\nabla_{\mathbf{p}_2} C(\mathbf{p}_1,\ldots,\mathbf{p}_4) &= -\frac{1}{|\mathbf{n}|}\left(\mathbf{e}_{14}\mathbf{M}\mathbf{e}_{34}^*\right)\\
\nabla_{\mathbf{p}_3} C(\mathbf{p}_1,\ldots,\mathbf{p}_4) &= -\frac{1}{|\mathbf{n}|}\left(\mathbf{e}_{14}\mathbf{M}\mathbf{e}_{12}^*\right)\\
\nabla_{\mathbf{p}_4} C(\mathbf{p}_1,\ldots,\mathbf{p}_4) &= \frac{1}{|\mathbf{n}|}\left(\mathbf{e}_{14}\mathbf{M}\mathbf{e}_{12}^*+\mathbf{n}\right)\\
\mathbf{M} &= \mathbf{I}-\frac{1}{|\mathbf{n}|^2}\mathbf{n}\mathbf{n}^T,
\end{aligned}$$

where $\mathbf{e}_{xy} = -\mathbf{p}_x+\mathbf{p}_y$ is the vector from point $\mathbf{p}_x$ to point $\mathbf{p}_y$ and $\mathbf{I}$ is the identity matrix. The operator $\mathbf{a}^*$ taking a vector $\mathbf{a}$ as argument creates the skew-symmetric matrix describing the cross-product as $\mathbf{a}^*\mathbf{b} = \mathbf{a}\times\mathbf{b}$. Using the gradients our solver computes position corrections (see Section 4.2.2) for the particles of the two edges.

We use a discrete collision detection in combination with a bounding volume hierarchy to find the collisions. In order to not negatively affect the performance we search for collisions only once before the constraint enforcement loop (see algorithm 1). Although particles are moved during the constraint solution step and may encounter new collisions we did not experience severe artifacts.

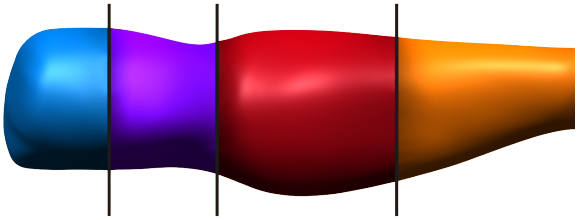


Figure 3: *The arm model is subdivided by intersecting planes into multiple sub-volumes to perform the volume conservation locally. The subdivision can be chosen freely by the artist.*

4.6. Volume Conservation

Volume conservation leads to two effects. On the one hand artifacts of LBS like deflating joint regions and the candy wrapper effect are avoided and on the other hand realistic bulging near self-colliding body parts is generated. We adopt the algorithm of Diziol et al. [DBB11] which suits well to our position-based simulation pipeline. In their work a single volume constraint is used for the whole deformable mesh. Without further weighting of the particles the mesh behaves like a balloon. As a result a volume loss at one region of the mesh will distribute globally through the whole mesh and increase the volume at every other region. Considering a virtual character pressing the foot will inflate the ears. This is usually not observable at real human beings. For this reason, Diziol et al. propose a weighting scheme of the mesh particles based on the total correcting displacement of the particle during the shape matching step. In our case this weighting scheme does not work well since our deformable mesh is attached to a skeleton which is static from the viewpoint of the skin simulation. Thus, the coupling constraints work against the shape matching constraints which will lead to high weights in the volume correction. In the next time step shape matching will not only work against the coupling constraints but also against the volume correction of the previous time step. As the constraint enforcement loop is terminated after a few steps, the interplay of different constraints might not find its equilibrium. In different experiments with our skinning model we saw severe artifacts when using the weighting scheme of Diziol et al.

To solve this problem we divide the mesh into sub-volumes instead of using one volume for the whole mesh (see Figure 3). The sub-volumes are located around joints where we want to capture volume changes due to self-collisions and deformations. To create the sub-volumes we first define cutting planes along the bones of the animation skeleton where the sub-volumes shall end. The normal of the plane is defined to be parallel to the corresponding bone segment. These planes intersect a triangle ring of the mesh around the bone. We search a connected path of edges in this ring of intersected triangles that lies in the cutting plane. In order to create a closed mesh for the sub-volume we search all triangles of the original mesh that lie between the connected paths of edges defining the sub-volume by a flood fill algorithm. To close the sub-volumes triangle-fans are created with the help of the edges in the connected paths. For every edge a triangle is created. The third point of the triangle is defined by the center point of the bone that lies on the cutting plane corresponding to the particular path. The artist is free to place the planes along the bones. In our examples we create sub-volumes for every bone by placing the cutting planes at the endpoints of the bones. In regions where there are several bones we only create one sub-volume. In case of the hand in Figure 5, for example, we moved the cutting planes to the middle of the proximal phalanges (first finger segments) and the carpal bone to create the hand sub-volume.

For every generated sub-volume mesh we create a volume constraint. In every time step these constraints are resolved after the constraint enforcement loop for the elastic model and collision constraints. This guarantees that the volume correction does not interfere with other constraints. As a result the volume is preserved even when the constraint enforcement solver does not converge in case of a limited iteration count. Since sub-volumes of neighboring joints share vertices, we solve the volume correction in a second loop. Generally a few iterations are sufficient to conserve the volume appropriately. The iterations allow neighboring constraints to find an equilibrium solution for the shared vertices. To avoid a violation of resolved collision constraints during the volume correction, we set the weights of colliding particles to zero, as proposed by Diziol et al. [DBB11].

5. Results

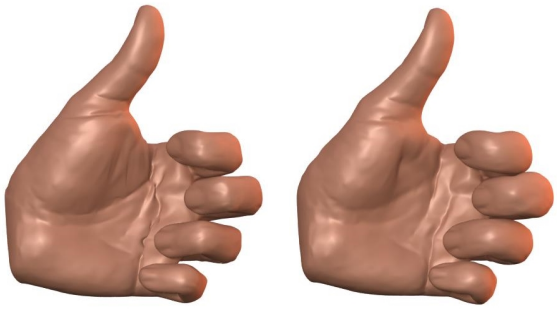
All experiments in this section were performed on an Intel Core i7-2600 with 3.4GHz, 8GB of memory and a GeForce 580 GTX. We use a fixed time step size of 5ms to capture all collisions with our discrete collision detection. In general, our time-stepping scheme is stable with clearly larger time step sizes. Our performance results are presented in Table 1. The performance of our method depends mainly on the simulation of elasticity. However, we used a high iteration count to compute a high quality deformation result near contact regions. Though, the artist can achieve a significantly higher performance at the cost of quality by performing shape matching only every n-th iteration. Nevertheless, even when solving the elastic constraints in every iteration, like done in our results, our method performs at interactive speed.

We demonstrate the deformation results of our approach with different models. The handling of the "elbow collapse" effect and self-collisions are shown in Figure 4. While the LBS results suffer under a volume loss of about 10 % for the whole mesh, our model successfully preserves the volume

Model	# vertices	$t_{Elasticity}$	t_{Volume}	t_{Total}	# iterations	Projection parameter			Stiffness s			
		[ms]	[ms]	[ms]		Fat	Muscle	Bone	Skin	Fat	Muscle	Bone
Arm	2916	68.4	1.8	77.5	20	0.15	0.45	0.8	0.8	0.6	0.7	0.8
Hand	5404	131.7	3.7	144.1	15	0.1	0.5	0.8	0.8	0.8	0.9	0.9
Big Man	8570	172.6	4.8	191.8	15	0.25	0.55	0.8	0.7	0.6	0.9	0.95

Table 1: *Timings of example scenarios. From left to right: the number of vertices, the average time of the elasticity loop, the average time of the volume correction loop, the average time of the total time step including collision detection, the number of iterations of the elasticity loop, the projection parameter for the fat, muscle, and bone layers, and the stiffness parameter s for each layer. While the projection and stiffness parameters are uniform for the Arm and the Hand model, the Big Man model uses non-uniform parameters per bone. The table includes the values for the belly.*

and differs only by 0.9 % from the initial volume by iterating four times through the volume correction loop in every time step. In addition, the usage of distance constraints at the joints improves the representation of the bones next to the joint. Furthermore, our method generates a plausible contact region. Slight bulges arise around the contact area. More important, the deformation of the upper arm is only guided by the contact with the lower arm which is best seen in the accompanying video. On the contrary, using LBS skinning the upper arm deflates even when there is no contact with the lower arm. The advantage of reproducing the elasticity of internal tissue is shown in Figure 5. The creases in the hand after flexing the fingers and moving the thumb are reproduced plausible by our method. In contrast, the LBS skinning result contains several artifacts at the crease lines. Our third model shows the advantages of the multi-layer approach in a complete character animation (see Figure 6). The main problems of LBS skinning appear in the groin region during the walking cycle. While the character raises its thigh the groin deforms unnaturally. Finally, the thigh intersects the lower body. Further artifacts appear at the shoulders and to a minor extent between belly and chest. Our multi-layer skinning creates no artifacts at the thigh. Instead, it generates a plausible deformation of the groin and resolves the collision between thigh and lower body. Using our simulated skin the shoulder and belly deform smoothly and secondary motion effects appear at the belly (see the accompanying video).



(a) LBS (b) Multi-layer skinning

Figure 5: *Palmar creases after extending the thumb and flexing the proximal phalanges of the remaining fingers. Note the artifacts of LBS skinning, especially at the heart line. Conversely, our multi-layer skin model successfully reproduces the internal elasticity of the hand tissue and generates a plausible result.*

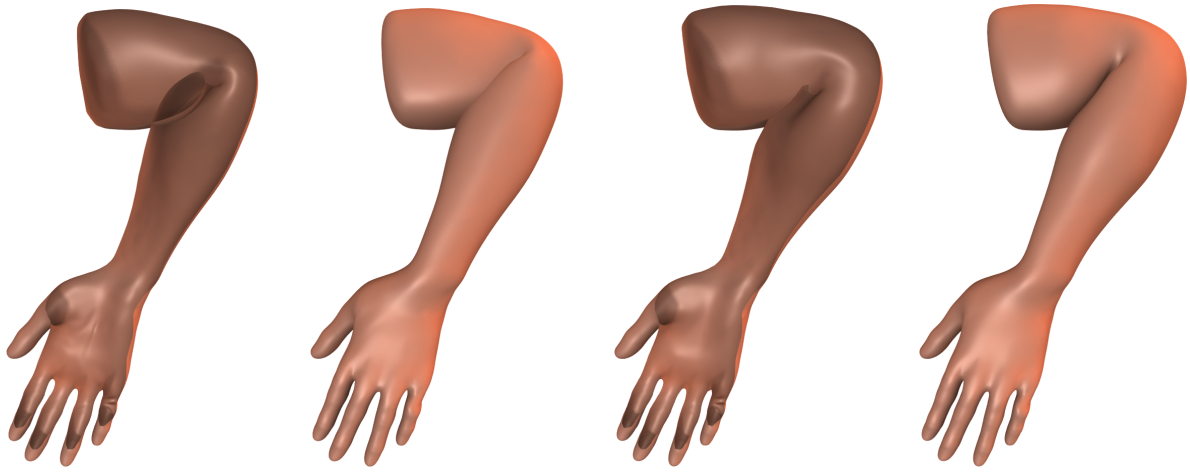
6. Conclusion

In this paper we introduced a novel multi-layer skin model for physically-based character skinning that supports secondary motion effects, collision handling and volume conservation. The meshes for all inner layers are determined automatically. Each layer of the model represents a different type of tissue. Hence, an artist can easily control the elastic behavior of muscles, fat and skin by simply adjusting the stiffness parameters of the corresponding layer. We use different position-based methods to simulate elasticity, handle collisions, conserve the volume and couple the skeleton with the deformable model. These methods allow a fast and unconditionally stable simulation.

In future we want to implement our physically-based skinning algorithm on the GPU. Since the fast summation technique is well-suited for the use on a graphics processor, we think that such an implementation will increase the performance significantly. Moreover, we plan to use a continuous collision detection since the discrete detection could miss collisions in case of fast motions. Finally, we want to integrate the skeleton in the simulation as articulated body [BETC12] and extend our method to simulate a two-way coupling between the skeleton and the skin model. This allows a more realistic motion since collision events and external forces influence the whole model and not only the skin layers.

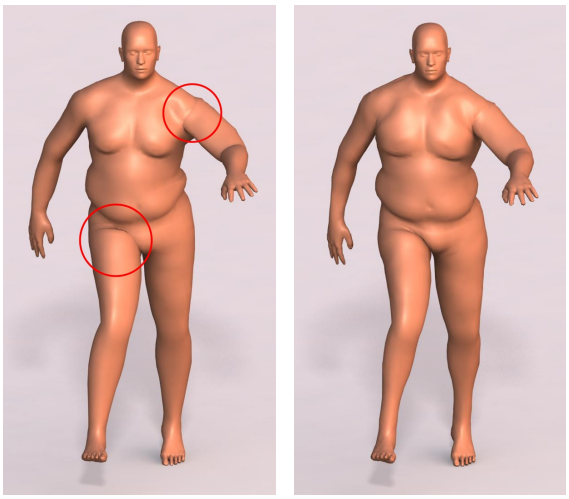
Acknowledgment

The work of Crispin Deul and Jan Bender was supported by the Excellence Initiative of the German Federal and State Governments and the Graduate School CE at TU Darmstadt.



(a) LBS (b) LBS (c) Multi-layer skinning (d) Multi-layer skinning

Figure 4: *A comparison of LBS skinning and our multi-layer approach after flexing the elbow joint of an arm model with 2916 vertices: (a) and (c) show a cut through the mesh while (b) and (d) show the skinning result. LBS skinning ((a) and (b)) suffers under interpenetration and volume loss at the elbow joint. In contrast, our approach ((c) and (d)) handles the self-collision of upper and lower arm successfully and preserves the volume.*



(a) LBS (b) Multi-layer skinning

Figure 6: *Compared to LBS, our multi-layer model generates plausible deformations at the groin and the shoulder without intersections (see red circles).*

References

[BETC12] BENDER J., ERLEBEN K., TRINKLE J., COUMANS E.: Interactive simulation of rigid body dynamics in computer graphics. In *EUROGRAPHICS 2012 State of the Art Reports* (2012), Eurographics Association. 8

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21*, 3 (July 2002), 594–603. 2, 6

[BMOT13] BENDER J., MÜLLER M., OTADUY M. A., TESCHNER M.: Position-based methods for the simulation of solid objects in computer graphics. In *EUROGRAPHICS State of the Art Reports* (2013), Eurographics Association. 2

[BP07] BARAN I., POPOVIĆ J.: Automatic rigging and animation of 3d characters. *ACM Trans. Graph. 26*, 3 (July 2007). 3

[CBC*07] CAPELL S., BURKHART M., CURLESS B., DUCHAMP T., POPOVIĆ Z.: Physically based rigging for deformable characters. *Graph. Models 69*, 1 (2007), 71–87. 3

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

[CLTL11] CHEN C.-H., LIN I.-C., TSAI M.-H., LU P.-H.: Lattice-based skinning and deformation for real-time skeleton-driven animation. In *Proc. Computer-Aided Design and Computer Graphics* (Sept. 2011), pp. 306 –312. 2, 3

[DBB09] DIZIOL R., BENDER J., BAYER D.: Volume conserving simulation of deformable bodies. In *Proceedings of Eurographics* (Munich (Germany), Mar. 2009). 2

[DBB11] DIZIOL R., BENDER J., BAYER D.: Robust real-time deformation of incompressible surface meshes. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2011), ACM, pp. 237–246. 5, 7

[GOT*07] GALOPPO N., OTADUY M. A., TEKIN S., GROSS M., LIN M. C.: Soft articulated characters with fast contact handling. *Computer Graphics Forum 26*, 3 (2007), 243–253. 3

[ISF07] IRVING G., SCHROEDER C., FEDKIW R.: Volume conserving finite element simulations of deformable models. *ACM Trans. Graph. 26*, 3 (July 2007). 2

[Jak01] JAKOBSEN T.: Advanced character physics. In *Proceedings of the Game Developer's Conference 2001* (2001). 6

[JBK*12] JACOBSON A., BARAN I., KAVAN L., POPOVIĆ J., SORKINE O.: Fast automatic skinning transformations. *ACM Trans. Graph. 31*, 4 (July 2012), 77:1–77:10. 2

[KCvO08] KAVAN L., COLLINS S., ŽÁRA J., O'SULLIVAN C.: Geometric skinning with approximate dual quaternion blending. *ACM Trans. Graph. 27*, 4 (Nov. 2008), 105:1–105:23. 1, 2

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

[KP11] KIM J., POLLARD N. S.: Fast simulation of skeleton-driven deformable body characters. *ACM Trans. Graph. 30*, 5 (Oct. 2011), 121:1–121:19. 2, 3

[KS12] KAVAN L., SORKINE O.: Elasticity-inspired deformers for character articulation. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 196:1–196:8. 2

[LCF00] LEWIS J. P., CORDNER M., FONG N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proc. SIGGRAPH* (2000), pp. 165–172. 2

[MC11a] MÜLLER M., CHENTANEZ N.: Adding physics to animated characters with oriented particles. In *Virtual Reality Interactions and Physical Simulations* (2011), Eurographics Association, pp. 83–91. 6

[MC11b] MÜLLER M., CHENTANEZ N.: Solid simulation with oriented particles. *ACM Trans. Graph. 30*, 4 (July 2011), 92:1–92:10. 4, 5, 6

[MDSB03] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*. Springer-Verlag, 2003, pp. 35–57. 4

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (Apr. 2007), 109–118. 4, 6

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (2005), 471–478. 4

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

[MTLT88] MAGNENAT-THALMANN N., LAPERRIÈRE R., THALMANN D.: Joint-dependent local deformations for hand animation and object grasping. In *Proc. Graphics Interface* (1988), Canadian Information Processing Society, pp. 26–33. 1, 2

[MZS*11] MCADAMS A., ZHU Y., SELLE A., EMPEY M., TAMSTORF R., TERAN J., SIFAKIS E.: Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph. 30*, 4 (July 2011), 37:1–37:12. 2, 3

[PH08] PARK S. I., HODGINS J. K.: Data-driven modeling of skin and muscle deformation. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 96:1–96:6. 1, 2

[RJ07] RIVERS A. R., JAMES D. L.: Fastlsm: fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph. 26*, 3 (July 2007). 2

[TK09] TAKAMATSU K., KANAI T.: Volume-preserving LSM deformations. In *ACM SIGGRAPH ASIA 2009 Sketches* (2009), pp. 15:1–15:1. 2

[TMOT12] TANG M., MANOCHA D., OTADUY M. A., TONG R.: Continuous penalty forces. *ACM Trans. Graph. 31*, 4 (July 2012), 107:1–107:9. 2

[TT93] TURNER R., THALMANN D.: The elastic surface layer model for animated character construction. In *Proc. Computer Graphics International* (1993), Springer-Verlag, pp. 399–412. 3

[TW91] TERZOPOULOS D., WATERS K.: Techniques for realistic facial modeling and animation. In *Computer Animation*, Thalmann N., Thalmann D., (Eds.). 1991, pp. 59–74. 3

[VBG*13] VAILLANT R., BARTHE L., GUENNEBAUD G., CANI M.-P., ROHMER D., WYVILL B., GOURMEL O., PAULIN M.: Implicit skinning: real-time skin deformation with contact modeling. *ACM Trans. Graph. 32*, 4 (July 2013), 125:1–125:12. 2

[vFTS08] VON FUNCK W., THEISEL H., SEIDEL H.-P.: Volume-preserving mesh skinning. In *Proc. VMV* (2008), pp. 409–414. 2