Technical Report TR-NCCA-2009-01

FAST SIMULATION OF SKIN SLIDING

Xiaosong Yang, Richard Southern and Jian Jun Zhang

Bournemouth
University

The National Centre for Computer Animation
Bournemouth Media School
Bournemouth University
Talbot Campus,
Poole, Dorset BH12 5BB
United Kingdom
2009

Technical Report TR-NCCA-2009-01

ISBN 1-85899-123-4

Title Fast simulation of skin sliding

Author(s) Xiaosong Yang, Richard Southern and Jian Jun Zhang

Keywords Deformation, Skin sliding

Abstract

Skin sliding is the phenomenon of the skin moving over underlying layers of fat, muscle and bone. Due to the complex interconnections between these separate layers and their differing elasticity properties, it is difficult to model and expensive to compute. We present a novel method to simulate this phenomenon at real–time by remeshing the surface based on a parameter space resampling. In order to evaluate the surface parametrization, we borrow a technique from structural engineering known as the force density method which solves for an energy minimizing form with a sparse linear system. Our method creates a realistic approximation of skin sliding in real–time, reducing texture distortions in the region of the deformation. In addition it is flexible, simple to use, and can be incorporated into any animation pipeline.

Report date 30 March 2009

Web site to download from http://eprints.bournemouth.ac.uk/

The authors' e-mail addresses

{xyang,rsouthern,jzhang}@bmouth.ac.uk

Supplementary notes

To appear *Computer Animation and Virtual Worlds*

The National Centre for Computer Animation
Bournemouth Media School
Bournemouth University
Talbot Campus,
Poole, Dorset BH12 5BB
United Kingdom

Fast simulation of skin sliding

Xiaosong Yang, Richard Southern and Jian Jun Zhang*

Monday 30th March, 2009

Abstract

Skin sliding is the phenomenon of the skin moving over underlying layers of fat, muscle and bone. Due to the complex interconnections between these separate layers and their differing elasticity properties, it is difficult to model and expensive to compute. We present a novel method to simulate this phenomenon at real–time by remeshing the surface based on a parameter space resampling. In order to evaluate the surface parametrization, we borrow a technique from structural engineering known as the force density method which solves for an energy minimizing form with a sparse linear system. Our method creates a realistic approximation of skin sliding in real–time, reducing texture distortions in the region of the deformation. In addition it is flexible, simple to use, and can be incorporated into any animation pipeline.

1 Introduction

It is well known in the computer animation industry that *secondary animation* greatly enhances the realism of a scene. They consist of subtle animations such as rippling of cloth or jiggling of muscles, which are often either smoothed away by motion capture systems, or not captured at all. For this reason, these complex animations are typically added by hand by a skilled animator. The automation of these tedious tasks has become an active area of research in computer graphics.

Skin sliding is a typical example of secondary motion. Due to the large number of physical properties which are attached to it, the movement of skin over the body is more dynamic than kinematic. It overlays multiple layers of underlying anatomical structures, such as fat, muscle and bone. In dermatological terms the skin is the largest organ of the integumentary system — an organ system which protects the body from harm — and is made up of multiple layers of epithelial tissues that guard the underlying muscles and organs. The movement of skin during body motion is highly dependent on all of these complex components, making it essential for any viable simulation of this phenomenon to draw upon the physical properties of the human body.

Incorporating physical properties of anatomy structures has already been shown to improve realism. Physics has been used to simulate the behaviour of muscles, bones, fat and the elastic skin layer. The skins elasticity has been approximated using mass spring systems which attach the skin surface to the underlying anatomical structure. As an example of such an application Stinson and Thuriot [1] demonstrated a physically based simulation used in the movie *Hellboy*. The developers built a layer based anatomical system comprising of bones, muscles, fat and

*Correspondence to: Jian J. Zhang, The Media School, Bournemouth University, Poole BH12 5BB, UK. *jzhang@bmth.ac.uk*.

skin to enhance the realism of the hellish creatures depicted in the movie. While the results are impressive, the computation is very expensive due to the large number of springs required for a realistic model. In addition, this complexity makes specifying the model very challenging for animators, and achieving a specific effect through tweaking spring parameters is particularly difficult.

In this paper we present a fast method to simulates the skin sliding phenomenon after deformation. The models before and after deformation are both embedded into a 2D parameter space. The elasticity of the skin is approximated at this stage by using the *force density method* for the embedding. The barycentric coordinates of the undeformed embedding in terms of the deformed embedding are deduced for each vertex. These are then used to interpolate the positions of these vertices on the deformed surface.

As our method is independent of the deformation technique used, it is easy to integrate into almost any animation pipeline. The method is simple to use — an animator only needs to specify the patch where the skin is permitted to slide and the sliding is entirely automatic. The method is also flexible, in that the force density of each edge can be specified individually by the animator. To our knowledge, this is the first method published which can automatically simulate the skin sliding phenomenon interactively.

2 Method Overview

We present a fast method to approximate the physical properties of sliding skin. An overview of our method is given in Figure 1.

Our method accepts two meshes as input $\mathcal{M}_0$, the original surface before deformation, and $\mathcal{M}_d$, the deformed surface mesh, where a mesh $\mathcal{M} = \{V, E, F\}$ consists of n vertices $v_i \in V$, m edges $e_i \in E$ and faces $f_i \in F$. The user defines a patch $\mathcal{K}_0$ on $\mathcal{M}_0$ which is allowed to slide. A mapping is constructed between the boundary of this region $\delta\mathcal{K}_0$ to the boundary in the parameter domain $\mathcal{U}_0$, $\delta\mathcal{U}_0$.

For a patch $\mathcal{K}_0 \in \mathcal{M}_0$ and the corresponding patch $\mathcal{K}_d \in \mathcal{M}_d$, use the *force density method* along with $\delta\mathcal{K}_0 \mapsto \delta\mathcal{U}_0$, $\delta\mathcal{K}_d \mapsto \delta\mathcal{U}_d$ to construct an embedding of these patches into parameter space $\mathcal{U}_0$ and $\mathcal{U}_d$ respectively.

We find the barycentric coordinates β_i of each vertex $u_i \in \mathcal{U}_0$ in the appropriate facets of $\mathcal{U}_d$. The positions of $v_i \in \mathcal{M}_d$ are updated by replacing them with vertices interpolated in the facets of $\mathcal{M}_d$ using each β_i. This resampling produces the effect of the vertices moving across the surface, and hence the skin sliding over layers of muscle and fat.

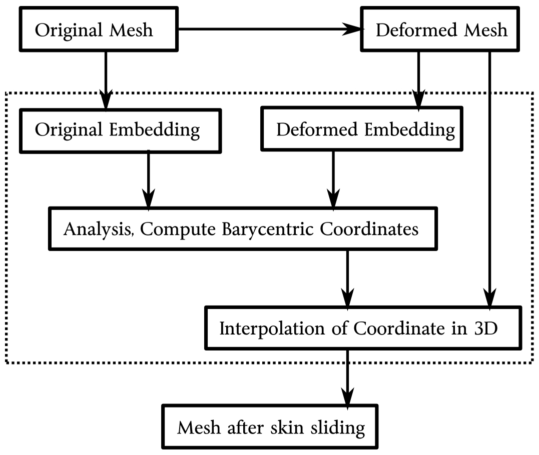


Figure 1: An overview of our method.

3 Motivation

Our approach can be easily motivated by means of a simple example. For the skin surface to slide, a point on the skin surface in close proximity to the underlying anatomical structure should remain approximately stationary when the underlying structure moves. This is demonstrated in Figure 2. After the sphere has moved under the surface of the skin, if there is no correction of the vertex locations, the skin is stretched on one side and compressed on the other (as it is in (b)). This results in poor surface sampling, which can affect surface properties such as texture coordinates. In (c) the surface is resampled so as to correct this error.

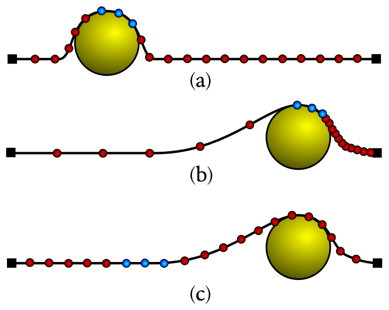


Figure 2: A demonstration of the desired skin sliding behaviour.

It has been suggested that an alternative method to prevent texture distortion would be to simply resample the texture coordinates rather than the geometry itself. Unfortunately this is very difficult in practice, as texture maps typically consist of many disconnected texture patches, and the behaviour of skin sliding at the boundary of one patch is undefined, presumably leading to unsightly seams, discontinuities and distortion.

4 Related work

In this section we will discuss related work in surface and skin deformation, as well as surface parametrization techniques that are directly applicable to the method presented in this paper.

4.1 Deformation techniques

Free Form Deformation (FFDs), first introduced by Sederberg and Parry [2], remains popular and have been adopted by many animation packages due to its simplicity and performance. Despite significant developments (Coquillart [3], Hsu et al. [4], MacCracken and Joy [5], Singh and Kokkevis [6]), these techniques have not been used to simulate the behavior of the physical properties of virtual characters.

A popular method of performing skin deformation, first introduced by Catmull [7], is to associate the skin surface with its underlying skeleton. The skin is treated as a shell that is deformed by a weighted combination of joint transformations applied to the character's skeleton. Collectively we will refer to such methods as the Smooth Skinning, although they are known by many nomenclatures, including sub-space deformation (SSD), linear blend skinning and enveloping. Many papers have since extended this approach (Lewis et al. [8], Magnenat-Thalmann et al. [9], Wang and Phillips [10], James and Twigg [11], Kavan et al. [12]).

Its popularity stems from its interactivity and simplicity of specifying animations by joint angles. As a result, it is an integral and established component of the current animation work flow.

Animators typically simulate muscle jiggle and skin sliding by using *influence objects* to drive the deformation. These are essentially additional "bones" (as described above) to which the skin vertices are bound. These give the animator more control over how the skin moves on the surface with respect to the skeletal animation, as these bones may move independently of the underlying skeleton structure. Unfortunately this process is entirely manual — the animator must place the influence object, attach it to the skin with skinning weights, and define how motion of the influence object translates to the motion of the skin. In addition, once these weights are painted, the skin vertex remains bound to the influence object, making future skin deformations difficult without first detaching the skin from the influence object.

4.2 Physically based approaches

Methods for modelling skin deformations can be separated into two main approaches. Physics or anatomy based approaches[13;14] and shape example based approaches such as blend shapes or the method of Weber et al. [15]. Physical methods are based on models of the human body incorporating elastic mechanics or biomechanics of skin deformations originating from the movements of muscles and tendons. While providing realistic simulations, these approaches are very slow to compute and difficult to use.

Turner and Thalmann [16] present an approach using a spring based elastic surface layer model. It uses a layered structure of anatomical parts from the inside out, skeleton $\rightarrow$ bone $\rightarrow$ fat $\rightarrow$ skin. Venkatraman et al. [17] extend this idea to the simulation of wrinkles and skin sliding. The skin is deformed by anchoring the skin with anchor springs and modelling the restoring forces when the skin is displaced.

Aubel and Thalmann [18] describe a method to simulate muscle shape deformation. They create a muscle model which is classified into two layers — the action line and the surface mesh. Complex dynamics and collisions are possible in their system, providing a visually realistic output. The computation of this approach is expensive, and it is therefore used mainly for production quality visual effects.

Due to the considerable computation required to solve physically based deformations, commercial animation software has not included this functionality until very recently. Autodesk Maya [19] offers a muscle package which allows animators to paint the sliding weight in the muscle simulation. However, this function only pushes the skin surfaces outwards when there is a collision with the underlying muscle, ignoring skin elasticity and potentially causing surface distortion. Our method can be combined with this method to incorporate an elasticity property to the skin deformation which will give rise to more natural skin sliding effects about muscle and bone. Some examples of this are shown in Figure 14.

Example based approaches offer an alternative to physical based approaches for real–time computations. An artist can model several key poses, and new poses can be synthesized from these. Wang and Phillips [10] use multi–weight enveloping and statistical learning to improve this effect. Lewis et al. [8] implement an example–based mesh deformation method by using radial basis functions to interpolate between shapes. Allen et al. [20] extend this technique for use with arbitrary unrelated examples, such as laser range scanning data. As all possible skin sliding deformations are very difficult to predict when designing examples, it is very difficult to use examples to model this phenomenon.

4.3 Surface parametrization

Surface parametrization is the process of "unwrapping" a 3D model, providing a mapping from (x, y, z) space to the 2D (u, v) plane. This process is particularly useful for defining texture coordinates, but has numerous other applications. For the remainder of this paper, we will refer to this process as *embedding*, represented by the $\mathsf{embed}()$ function.

Numerous surface embedding methods exists, each fulfilling application–specific requirements, and can be classified by the geometric properties they try to preserve. These properties include edge lengths, facet area or angles. In addition, the boundary may be fixed or free. For further details on these techniques, the reader is referred to the review of Floater and Hormann [21].

Piponi and Borshukov [22] introduce a technique called *pelting* for finding an optimal texture mapping over subdivision surfaces. The user interactively inserts seams into the subdivision surface and attaches dynamic springs to connect the seam to the surrounding 2D rectangular frame. The edges around the cut form a topological circle. Vertices on this circle are connected to the corresponding points on the frame with a connecting spring. The stiffness of each spring and the spacing between the connecting points on the frame is proportional to the spacing between the opposite ends of the springs in the original model. A spring is then instantiated along each edge in the original model with stiffness proportional to the edge length. Solving this mass spring network yields an optimal embedding, but due to the oscillatory nature of these systems the results are typically not interactive.

The method presented in this paper can also be thought of as a pelting technique. However, by approximating the mass spring solution with the Force Density Method, we can significantly reduce the cost of computing the parametrization.

5 The Force Density Method

The Force Density Method (FDM), first introduced by[23], is commonly used in engineering to find the equilibrium shape of a structure consisting of a network of cables with different elasticity properties when stress is applied. While shape analysis of tensile structures is a geometrically non–linear problem, the FDM linearises the form–fitting equations analytically by using the *force density ratio* for each cable element, $q = F/L$, where F and L are the force and length of a cable element respectively.

Given the positions of nodes (vertices) which connect the cables (edges) of our network V, the topology of this network is encoded in the *branch–node matrix* C (see Figure 3. Given a load vector R and the diagonal matrix of force densities Q, the equilibrium location can be deduced by solving for X in

$$(C^T QC)X = R. \tag{1}$$

The FDM has certain properties which makes it of interest in computer graphics:

- it represents a *minimum energy surface*, and therefore
- it approximates a C^2 continuous surface,
- it depends only on the force density of the edges and the topology of the network, and
- the system is sparse, symmetric and positive definite, quickly solved using the conjugate gradient method.

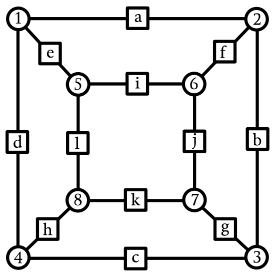
5.1 Embedding with FDM

In order to use the FDM for 2D embedding, we must be able to constrain nodes on the boundary. For this, the matrix C is separated into two sub–matrices: C_f contains constrained nodes with corresponding position X_f, while C contains those that are free to move. The problem in Equation 1 is reformulated as

$$(C^T QC)X = R - (C^T QC_f)X_f. \tag{2}$$

Note that for the purposes of embedding, R is typically set to zero.

The FDM is *fold–over free*. This is explained with reference to Figure 4. As the natural rest internal force load of each node is 0, any foldover will result in external forces applied to the nodes which have folded over. As a result it will not be in a state of equilibrium, and this configuration cannot occur.



	C_f				C			
	1	2	3	4	5	6	7	8
a	1	-1	.	.	.	.	.	.
b	.	1	-1	.	.	.	.	.
c	.	.	1	-1	.	.	.	.
d	1	.	.	-1	.	.	.	.
e	1	.	.	.	-1	.	.	.
f	.	1	.	.	.	-1	.	.
g	.	.	1	.	.	.	-1	.
h	.	.	.	1	.	.	.	-1
i	.	.	.	.	1	-1	.	.
j	.	.	.	.	.	1	-1	.
k	.	.	.	.	.	.	1	-1
l	.	.	.	.	1	.	.	-1

Figure 3: Deriving the branch–node matrix. Each row corresponds to an edge, while columns correspond to nodes. Note that the order in which the nodes are specified on the edge will not affect the system. Below, a head with internal boundaries is embedded into the quadrilateral domain using the force density method.

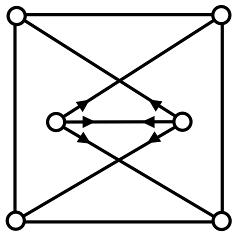
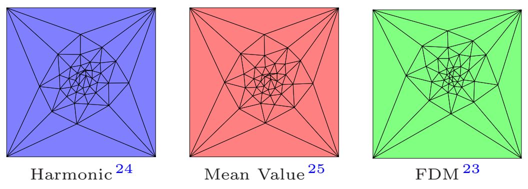


Figure 4: Foldover with FDM cannot occur, as it will not be in a state of equilibrium.



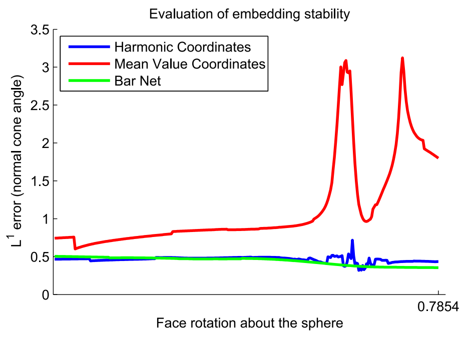


Figure 5: A comparison of the stability under motion of three embedding techniques.

5.2 Stability under motion

The FDM cannot guarantee any of the commonly advocated properties of embeddings in computer graphics, such as *isometry* (length preserving) or *conformality* (angle preserving). In our application, it is desirable that the path of a vertex or a group of vertices moving across the surface of the shape is mirrored in the embedding. We evaluate this phenomenon by measuring the distortion of these displacement vectors in the embedded space. We call this property *stability under motion.*

In Figure 5, we compare two popular fixed boundary conformal techniques, Harmonic mappings[24] and Mean Value Coordinates[25] with the FDM with unit edge forces. The stability under motion of these embedding methods is highly non–linear, and so we evaluate each embedding technique experimentally as follows:

1. Compute the embedding $\mathcal{U}_0 = \mathsf{embed}(\mathcal{M}_0)$.
2. Rotate a set of points on the sphere (in this case, one triangle) in a straight path around the surface of the sphere.
3. For each state of the rotation $\mathcal{M}_i$ compute $\mathcal{U}_i = \mathsf{embed}(\mathcal{M}_i)$.
4. Compute the n displacement vectors in

the embedded space. So for each $u_{0,i} \in \mathcal{U}_0$ and $u_{d,i} \in \mathcal{U}_d$ define $\mathbf{d}_i$ as the vector between $u_{0,i}$ and $u_{d,i}$, and

$$D = [\mathbf{d}_1, \ldots, \mathbf{d}_n]^T .$$

5. We use the angle of the normal cone of these displacement vectors $\alpha = \max_{i,j} (\operatorname{acos}(\mathbf{d}_i \cdot \mathbf{d}_j))$ as the distortion error. For this experiment, we evaluate the distortion of only the points moving on the surface.

Figure 5 demonstrates that under all rotations the FDM embedding is stable, even when faces of $\mathcal{M}_d$ overlap. In addition, there is considerably less displacement of surrounding nodes after rotations.

6 Finding patch boundaries

For practical purposes, the user is unlikely to specify that the entire skin surface is to slide due to the deformation. Certain areas of the body, for example have a more rigid attachment with the underlying fat and muscle. For this reason we require the user to specify a surface patch $\mathcal{K}_0 \in \mathcal{M}_0$ which is to be used for skin sliding (see Figure 6).

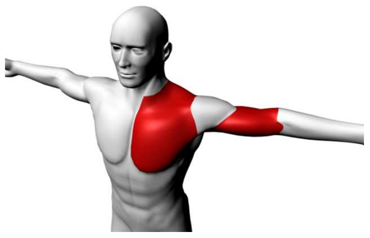


Figure 6: An example of quadrilateral patch selection.

In order to minimize distortion after embedding, the boundary of the surface patch should approximate the shape of the embedding space as much as possible. In this paper, we embed patches into the unit square, i.e. $\mathcal{U} \in [0,1]^2$, and therefore a reasonably quadrilateral 3D patch would yield best results. However as the embedded space can have any fixed boundary shape, the boundary shape could be defined based on the users selection.

A patch $\mathcal{K}$ may have internal boundaries, for example the holes in the head model in Figure 7 for eyes, nostrils and ears. These must also be flagged as boundary vertices and mapped to specified locations in $\mathcal{U}_0$. This is to prevent the method from resampling in a region in which there is no data, which may occur if boundary vertices of $\mathcal{U}_0$ and $\mathcal{U}_d$ are not fixed in the same location.

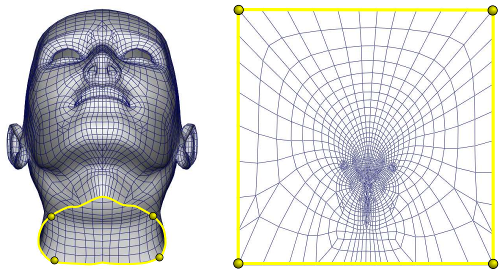


Figure 7: In this example, the head with internal boundaries is embedded into the quadrilateral domain using the force density method.

The selected corner vertices are fixed to the corners of the square. The locations of the remaining boundary vertices in the embedded space are deduced by using the proportional lengths on the original mesh of these boundary vertices from the corner vertices.

7 Embedding the patch

Once a patch $\mathcal{K}$ and its boundary $\delta\mathcal{K}$ has been identified, the FDM is used to embed this shape into the parametric domain. The FDM is flexible in that the force density for each edge must be defined for the matrix Q in Equation 2. Conceptually, the greater the force density, the shorter the edge will be in the embedding. Simply defining a unit force for each edge $e_i \in E$ is sufficient for most applications, i.e. $q_i = 1/|e_i|$.

A significant advantage of the FDM is that since our two patches $\mathcal{K}_0$ and $\mathcal{K}_d$ have the same connectivity and constraints, the matrices C, C_f and X_f are precomputed and reused for all proceeding embeddings. Only the force density Q needs to be updated for each subsequently deformed patch $\mathcal{K}_d$. We exploit this property to achieve interactive rates in our results.

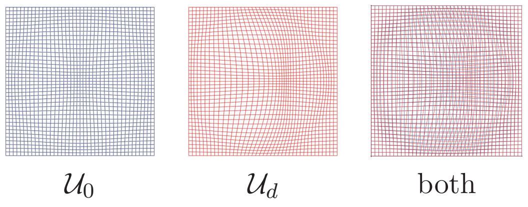


Figure 8: A comparison of undeformed and deformed embeddings of a regular grid.

For performance reasons we approximate the force density with

$$q_i = \max_j |e_j|^2 - |e_i|^2 + \epsilon,$$

where ϵ is a very small value used to prevent degenerate cases.

It is also possible for the user to specify the force density of each edge with a tool similar to smooth skinning weight painting. We have not implemented this feature, as specify the forces on individual edges may overcomplicate the design process.

8 Barycentric mapping and interpolation

In order to reposition the vertices of the deformed patch $\mathcal{K}_d$ to reflect the motion of skin over the surface, we must deduce the barycentric location of each point of the original patch $\mathcal{K}_0$ in the deformed patch $\mathcal{K}_d$, given the corresponding embeddings $\mathcal{U}_0$ and $\mathcal{U}_d$ respectively. This process involves two steps:

- we find the face $f_k \in \mathcal{U}_d$ in which each vertex $u_i \in \mathcal{U}_0$ lies, and then
- we find the barycentric coordinates β_i of the deformed vertex u_i in the face $f_k \in \mathcal{U}_d$.

In order to improve the performance of the point–in–face test, we first render the deformed embedding $\mathcal{U}_d$ to an lookup texture, with each facet rendered with a different colour. The point in triangle test is simply a case of using a colour lookup table to find the closest pixel to each vertex $u_i \in \mathcal{U}_0$. In practice, we will consider all referenced triangles in a 3×3 grid surrounding the closest pixel to ensure that edge cases are found. (see Figure 9).

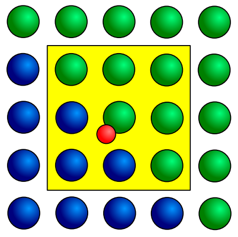


Figure 9: A method to find the face enclosing a vertex. Initially the embedding is rendered as a texture with each face as a different colour. The red vertex is the position of a vertex to evaluate. Any polygons in the 3×3 grid about the closest pixel to this vertex are tested.

For the examples used in this paper, a 1024^2 texture was sufficient to sample all triangles, as no pixel query ever resulted in more than 2 faces. As the size of the lookup texture will effect accuracy, it should be proportional to the number of nodes in patch $\mathcal{K}_d$.

In order to find the barycentric coordinates β_i of the point u_i in each of the returned faces, either barycentric[26] or generalized barycentric coordinates[27;28] can be used. In order to ensure interactive framerates, the surface is first triangulated and standard barycentric coordinates are used. Note also that using generalized barycentric coordinates are likely to result in non–planar facets.

For performance reasons, the interpolated vertex is simply linearly interpolated using the corner vertices of f_k and the barycentric coordinates β_i. This approach may result in changes to the enclosed volume of the surface. However, given that the deformation caused by skin sliding is very small this artefact is rarely noticeable. More sophisticated smooth interpolation techniques should be used when the quality of the rendered result is essential.

9 Results

Our system was implemented as a plug–in for Maya 2009 due to the extensive array of existing user interface options and query tools. The computer configuration for these experiments was an AMD Athlon 4200+ with 2GB RAM and NVidia GeForce 8500 GT. In Table 10 we show the absolute computation time and the number of iterations of the conjugate gradient method required to compute the embeddings of $\mathcal{K}_0$ and $\mathcal{K}_d$.

Many of the matrices in Equation 2 are precomputed in the computation of the embedding $\mathsf{embed}(\mathcal{K}_0)$. This explains why the computation of $\mathsf{embed}(\mathcal{K}_d)$ is real–time. The lookup texture is computed for each frame, but dramatically improves the performance of the point–in–triangle test used in the interpolation stage. Note that the computational cost of computing the barycentric coordinates and the interpolation is negligible with our timings, never taking longer than 1ms for all examples. The performance in FPS is approximated from the timings measured from the stages of our algorithm, and does not include the overhead due to rendering or other deformation computations.

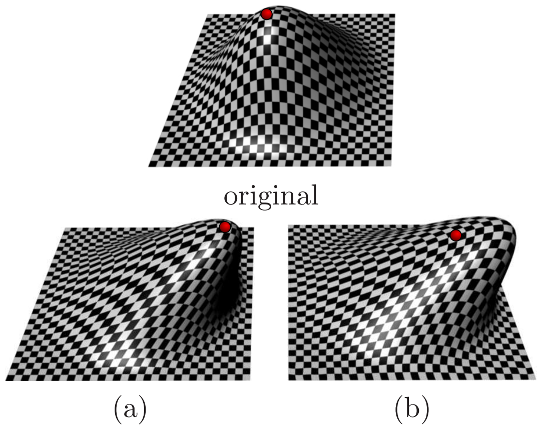


Figure 11: Skin sliding with a chequerboard texture on a regular grid. The highlighted point is the same on each mesh. In (a) the mesh is deformed with only a horizontal movement. In (b) our skin sliding method is applied to the same deformation, after which the chequerboard pattern becomes more regular, giving the impression that the point on the surface has remained stationary.

We include several results demonstrating the efficacy of our method. The checkerboard example (Figure 11) demonstrates the effect that our reparametrization has on texture mapped surfaces, effectively a 3D analogy of Figure 2. This effect is also demonstrated in a real–world application with the elbow (Figure 13) and tattoo sliding examples (Figure 12). The finger and elephant examples if Figure 14 illustrate the effect of our technique on animating surfaces.

10 Conclusions

Skin sliding is an effective secondary animation technique which is notoriously difficult to model realistically due to the complex anatomical dependencies of the outer layers of the human body. While mass spring systems may yield realistic results, they are at high cost, both in terms of computation time and the

	vertices	edges	embed($\mathcal{K}_0$)	embed($\mathcal{K}_d$)	Lookup Tex	FPS
elbow	51	90	3	0	1	1000
bicep	163	299	18	4	1	200
face	316	593	53	8	2	100
elephant	431	1218	110	15	3	55
finger	538	1060	153	12	2	72

Figure 10: Timing results in milliseconds for the examples in this paper.

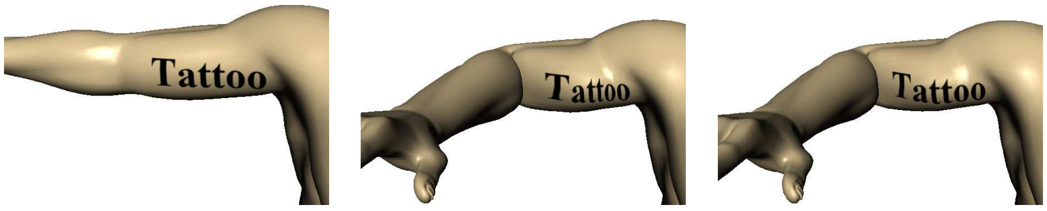


Figure 12: Skin sliding on the upper arm. In (a), the skin deforms without sliding correction, causing distortion of the texture. With skin sliding (b) the texture retains its shape, giving the impression of underlying motion.

difficulty of specifying a highly sophisticated system.

In this paper we have proposed a fast, simple and effective method to model skin sliding. In our system, the animator need only select a region of the surface which is permitted to slide. The boundary of a parametric domain is deduced from this selection, and the selected patch region is embedded into this domain. This process is repeated for each deformed frame of an animation. The skin sliding action is created by remeshing the deformed model based on the comparison of the deformed and undeformed embeddings. Our system is independent of the deformation technique used, and can therefore be easily incorporated into any existing animation pipeline.

This approach is made possibly by using a technique from structural mechanics known as the force density method. The elasticity of the skin is simulated by specifying the force density on each edge of the model to be embedded. We have shown that this technique is more stable under motion than either Harmonic coordinates or Mean Value coordinates, making it better suited for this application than other fixed boundary parametrization techniques. In addition, this method gives a significant performance advantage when recomputing an embedding with the same connectivity, affording us with interactive computation times.

As with all interactive graphics techniques, realism is sacrificed in order to improve performance. This method ignores the underlying anatomical structure, deriving instead the elasticity of the skin surface from the patch selected by the artist and the embedding as a result of the force density method. The quality of the results may be improved by using alternative embedding techniques that minimize the distortion of the surface in the embedding, or by using irregular embedding boundaries.

There are several avenues for future work stemming from this technique. It is possible to support for skin overlapping patches with this system, although handling the blending

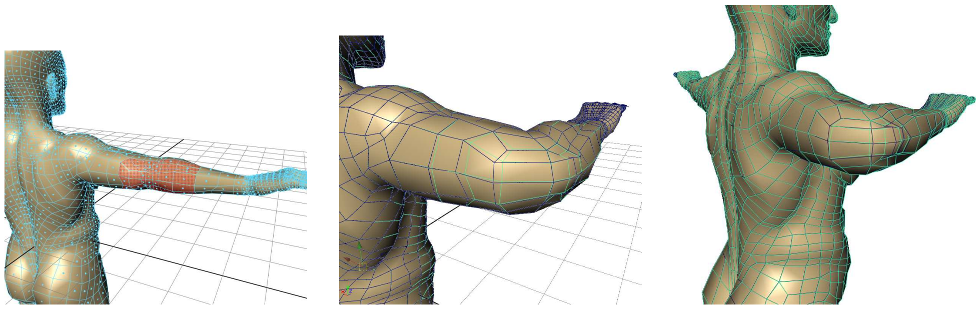


Figure 13: An example of using skin sliding on the elbow. This is a simple and effective usage of our technique.

between multiple resampling results may require significant user intervention. Wrinkling is a closely related secondary animation phenomenon, and incorporating a wrinkling simulation would be particularly useful. We also hope to extend the user interface to make the specification of patch boundaries and edge forces more intuitive. In addition, the performance of this method could conceivably be improved through further optimizations or implementation on graphics hardware, making this method applicable to real–time systems such as games.

11 Acknowledgements

Thanks go to Arun Somasekharan for his assistance. This research is in part supported by the Great Western Research Fellowship grant and EPSRC grant EP/F030355/1. All the models used for this paper are from 500 3D-Objects[29] and the Ultimate Human (http://www.cgcharacter.com/ultimatehuman.html).

References

[1] W. T. Stinson and P. G. Thuriot. Bulging muscles and sliding skin: Deformation systems for Hellboy. In *ACM SIGGRAPH 2004 Sketches*, page 65, 2004.

[2] T. Sederberg and S. Parry. Free–form deformation of solid geometric models. *Computer Graphics (Proceedings of SIGGRAPH)*, 20(4):151–160, 1986.

[3] S. Coquillart. Extended free-form deformation: A sculpturing tool for 3d geometric modelling. *Computer Graphics 24 (Proceedings of SIGGRAPH)*, 4:187–196, 1990.

[4] W. M. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. *Computer Graphics (Proceedings of SIGGRAPH)*, pages 177–184, 1992.

[5] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, pages 181–188, 1996.

[6] K. Singh and E. Kokkevis. Skinning characters using surface oriented free-form deformations. In *Graphics Interface*, pages 35–42, 2000.

[7] E. E. Catmull. A system for computer generated movies. In *Proceedings of ACM*

Annual Conference, pages 422–431, August 1972.

[8] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: a unified approach to shape interpolation and skeleton driven deformation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, pages 165–172, 2000.

[9] N. Magnenat-Thalmann, R. Laperrire, and D. Thalmann. Joint–dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics interface*, pages 26–33, 1988.

[10] X. Wang and C. Phillips. Multi-weight enveloping: leastsquares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 129–138, 2002.

[11] Doug L. James and Christopher D. Twigg. Skinning mesh animations. *ACM Transactions on Graphics (SIGGRAPH 2005)*, 24(3), August 2005.

[12] Ladislav Kavan, Rachel McDonnell, Simon Dobbyn, Jiří Žára, and Carol O'Sullivan. Skinning arbitrary deformations. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 53–60, 2007.

[13] J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. *Computer Graphics (Proceedings of SIGGRAPH)*, pages 243–252, July 1989.

[14] F. Scheepers, R. E. Parent, W. E. Carlson, and S. F. May. Anatomy–based modeling of the human musculature. *Computer Graphics (Proceedings of SIGGRAPH)*, pages 163–172, August 1997.

[15] O. Weber, O. Sorkine, Y. Lipman, and C. Gotsman. Context-aware skeletal shape deformation. *Computer Graphics Forum*, 26(3):265–274, 2007.

[16] R. Turner and D. Thalmann. The elastic surface layer model for animated character construction. In *Proceedings of CG International*, pages 399–412, 1993.

[17] K. Venkatraman, S. Lodha, and R. Raghavan. A kinematic-variational model for animating skin with wrinkles. *Computers & Graphics*, 29(5):756–770, October 2005.

[18] A. Aubel and D. Thalmann. Interactive modeling of the human musculature. In *Proceedings of Computer Animation*, pages 167–255, 2001.

[19] Autodesk Maya. Documentation. `http://download.autodesk.com/us/maya/2009help/index.html`, 2009.

[20] B. Allen, B. Curless, and Z. Popovic. Articulated body deformation from range scan data. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, pages 612–619, 2002.

[21] M. S. Floater and K. Hormann. *Advances in Multiresolution for Geometric Modelling, N. A. Dodgson, M. S. Floater, and M. A. Sabin (eds)*, chapter Surface Parameterization: a Tutorial and Survey, pages 157–186. Springer-Verlag, Heidelberg, 2004.

[22] D. Piponi and G. Borshukov. Seamless texture mapping of subdivision surfaces by model pelting and texture blending. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, pages 471–478, 2000.

[23] J. H. Schek. The force density method for form finding and computation of general networks. *Computer Methods in Applied Mechanics and Engineering*, (3):115–134, 1974.

[24] M. Eck, T. D. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. *Proceedings of SIGGRAPH*, pages 173–182, 1995.

[25] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1): 19–27, 2003.

[26] W. T. Tutte. How to draw a graph. *Proc. London Mathematical Society*, 13: 743–768, 1963.

[27] M. Meyer, H. Lee, A. Barr, and M. Desbrun. Generalized barycentric coordinates on irregular polygons. *Journal of graphics tools*, 7(1):13–22, 2002.

[28] K. Hormann and M. S. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Transactions on Graphics*, 25(4):1424–1441, October 2006.

[29] J. Wiedmann. *500 3D Objects*. Taschen Verlag, September 2002.

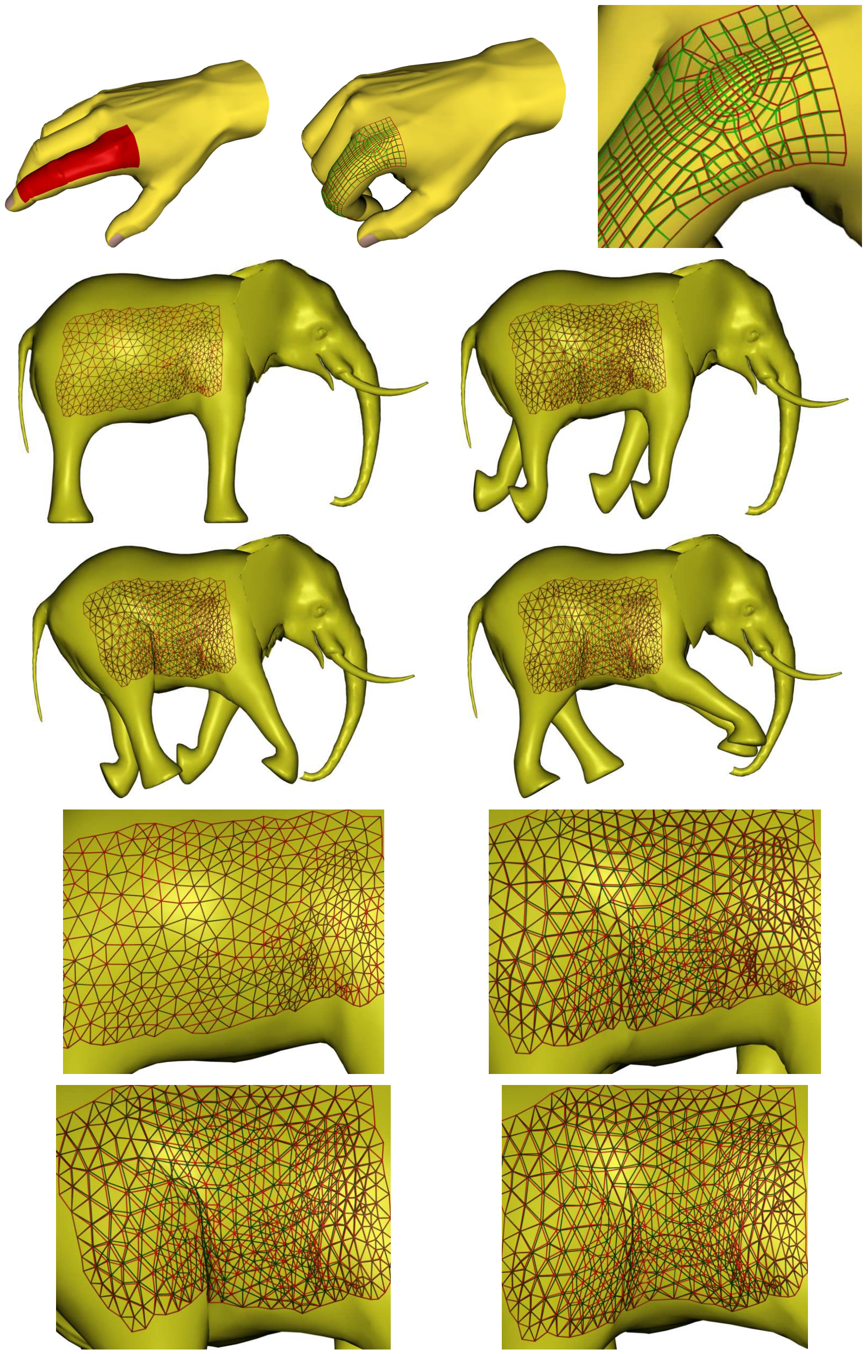


Figure 14: Several examples of skin sliding using our method. The original deformed mesh is shown in green, while the results of our technique are shown in red.