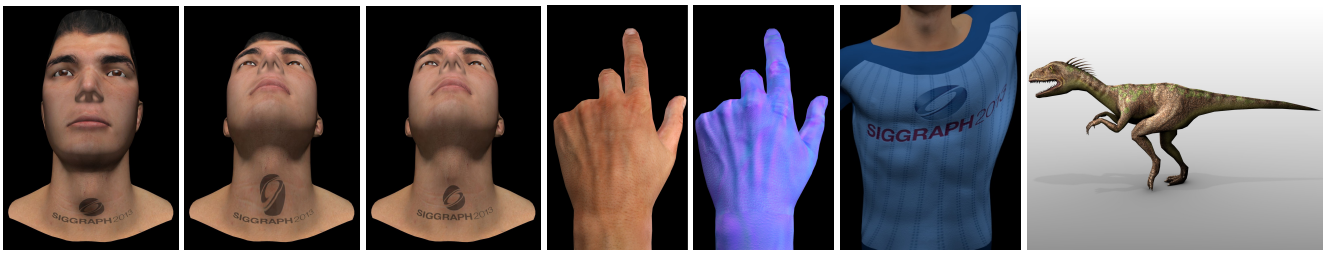
To appear in ACM TOG 32(4).

Thin Skin Elastodynamics

Duo Li Shinjiro Sueda* Debanga R. Neog Dinesh K. Pai

University of British Columbia



(a) (b) (c) (d) (e) (f) (g)

Figure 1: *The skin of a character (a) can be significantly distorted using standard techniques (b), but moves realistically with our method (c). Flexing a hand (d) realistically moves the skin, along with skin properties like normal maps (e). Our method can also be applied to skin-tight clothes (f) and animal skin (g).*

Abstract

We present a novel approach for simulating thin hyperelastic skin. Real human skin is only a few millimeters thick. It can stretch and slide over underlying body structures such as muscles, bones, and tendons, revealing rich details of a moving character. Simulating such skin is challenging because it is in close contact with the body and shares its geometry. Despite major advances in simulating elastodynamics of cloth and soft bodies for computer graphics, such methods are difficult to use for simulating thin skin due to the need to deal with non-conforming meshes, collision detection, and contact response. We propose a novel Eulerian representation of skin that avoids all the difficulties of constraining the skin to lie on the body surface by working directly on the surface itself. Skin is modeled as a 2D hyperelastic membrane with arbitrary topology, which makes it easy to cover an entire character or object. Unlike most Eulerian simulations, we do not require a regular grid and can use triangular meshes to model body and skin geometry. The method is easy to implement, and can use low resolution meshes to animate high-resolution details stored in texture-like maps. Skin movement is driven by the animation of body shape prescribed by an artist or by another simulation, and so it can be easily added as a post-processing stage to an existing animation pipeline. We provide several examples simulating human and animal skin, and skin-tight clothes.

CR Categories: I.6.8 [Simulation and Modeling]: Types of Simulation—Combined

Keywords: skin, faces, hands, physically-based simulation, constrained simulation, Eulerian simulation

* Now at Disney Research Boston

1 Introduction

Beauty, in computer animation, is often skin deep. It is the motion of skin that is finally seen in an animation and a poor skin model can ruin important details of movement. For example, consider the animation shown in Fig. 1a in which a character's surface mesh is carefully animated to look up and swallow. If one directly applies detailed skin textures to the mesh, the skin near the base of the neck does not move when the head is raised, and stretches unrealistically when the character's Adam's apple moves up to swallow, destroying the illusion of reality (Fig. 1b). Perhaps because of this problem, important details such as hairs, pores, veins, scars, and tattoos are often not included in the skin textures of moving characters. Using the method proposed here, the skin on the neck moves much more realistically (Fig. 1c).

Part of the problem is that the term "skin" in computer graphics is usually used to refer to the shape of a character's external surface, and all soft tissues beneath. To avoid confusion, we will refer to subcutaneous soft tissues such as muscles, fat, and tendons that give a character its 3D shape as the "body" and reserve the word "skin" to refer to the thin anatomical skin that covers the body.

This anatomical skin in humans and animals is a thin membrane, 2-3 mm thick in most areas, and is especially thin on visible areas such as the face and hands. Skin has evolved to be in close contact with the body and yet not impede movement by sliding and stretching over the body with little resistance.

Our goal is to capture these essential features of skin in an efficient and robust simulation. However, the close contact between skin and body presents serious difficulties for the usual methods used for simulating cloth and other thin structures. These methods discretize the thin structure into a mesh that has to interact with a separate mesh representing the body, presenting significant challenges for detecting and handling contact between these intimately osculating meshes.

We avoid all these problems by using a single mesh to represent

both the skin and the body. This results in a new Eulerian discretization of the dynamics of a hyperelastic membrane moving on a manifold. This is in contrast to the Lagrangian discretizations almost universally employed for cloth simulation. In the Eulerian setting the skin mesh is fixed on the body and can be identical to the body mesh. The skin is automatically constrained to move on the body and not separate from it, avoiding the need for contact processing. Even though the approach is unlike previous work in simulating skin and cloth, it shares many features with texture mapping and is easy to implement.

Contributions. We propose a general Eulerian discretization of hyperelastic membranes, such as skin, moving on a triangulated surface. The mesh can have arbitrary topology, which makes it possible to cover complete and complex characters. Any triangular mesh can be used, which makes it easy to collocate mesh vertices at constraints. We provide a general constraint handling method that serves as a flexible modeling tool for animators. The skin simulation is driven by body movement, and hence should be easy to integrate with existing animation pipelines as a post-processing stage.

2 Related Work

Because of its importance, a large body of research in computer animation is related to skin. We discuss only the closest work here. Due to space limitations we do not discuss work aimed at simulating generic solids, fluids, and cloth; but we refer to closely related techniques in the rest of the paper, in context. Broadly speaking, skin models can be classified as kinematic, example based, or physically based.

Kinematic skin. Perhaps the standard approach to skinning is to compute vertex positions of the skin mesh based on the configuration of the skeleton. The most widely used approach is linear blend skinning (or skeletal-subspace deformation) [Magnenat-Thalmann et al. 1988] and its refinements (e.g., [Kavan et al. 2008]).

Example based skin. Example based skinning is attractive since it can provide rich details from physical measurements [Huang et al. 2011; Beeler et al. 2011], animator input [Lewis et al. 2000; Mohr and Gleicher 2003], physical simulation [Kry et al. 2002], mesh animations [James and Twigg 2005], or motion capture [Park and Hodgins 2006]. The main difficulties with this approach are the complexity of acquiring the necessary data and poor generalization, especially to characters that only exist in our imagination.

Physically based skin. With few exceptions, previous work in this area focuses on simulating the deformation of soft tissues of the body and not on how skin moves on the body. Notable examples include generic soft tissues [McAdams et al. 2011], muscles [Teran et al. 2003], and tendons [Sueda et al. 2008]. These methods, as well as those listed earlier, are complementary to our work and could provide the detailed subcutaneous shapes on which our skin slides.

One of the most popular approaches to skin modeling is through passive mass-spring networks attached to muscles that drive the deformation. The seminal work of Terzopoulos and Waters [1990] on facial animation included deformable skin modeled with a mass-spring network. Similar approaches have been applied to simulating the skin of animals [Wilhelms and Gelder 1997], human hands [Albrecht et al. 2003], and for general texture mapping [Maillot et al. 1993]. More sophisticated finite element models have also been used. Membrane models have been used to simulate wrinkles [Wu et al. 1996], and for performance-driven animation [Choe et al. 2001]. Thin shell models [Qin and Terzopoulos 1996; Grinspun et al. 2003] extend membrane models by including resistance to bending; this could be useful for simulating thicker skin. Gourret et al. [1989], Sifakis et al. [2005] and Lee et al. [2009] model the skin as volumetric soft tissue along with muscles. Using a volumetric approach is advantageous or required for many applications, e.g., skin surgery [Sifakis et al. 2009].

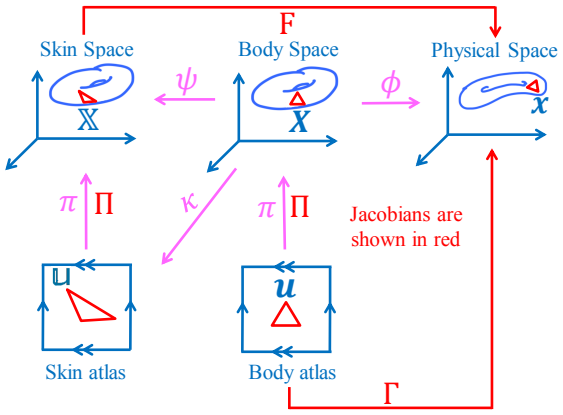


Figure 2: *Five spaces involved in representing elastic skin, and the mappings between them. See text for explanation.*

To our knowledge, no prior work in graphics has simulated skin or any thin elastic solid in the Eulerian setting that is the key to our method. The Eulerian setting is standard for simulating fluids, but has only recently been used for simulating solids—in 1D [Sueda et al. 2011] and in 3D [Levin et al. 2011; Fan et al. 2013]. Our method significantly extends this work to include 2D manifolds of arbitrary topology, allows arbitrary triangular meshes instead of regular grids, and integrates easily with standard animation pipelines.

3 Methods

3.1 Spaces

Simulating skin is challenging because skin is in close contact with the body and shares its geometry. Yet, the two are physically distinct. To carefully distinguish between skin and body movements we define the relevant spaces first. See Fig. 2. To make things concrete, it shows an example of elastic skin that completely covers the surface of a torus.

First look at the right hand side of the figure. *Physical space* is the familiar 3D space in which objects live; its points are denoted $\boldsymbol{x}$. *Body space* is the reference 3D space in which we embed the surface of the body upon which the skin slides, at time $t = 0$; points on the surface are denoted $\boldsymbol{X}$. We assume that the surface is represented as a triangle mesh, for generality. A triangle of the mesh is shown in the figure. While meshes are convenient, it is useful to also have a parameterization of the surface. *Body atlas* is a collection of 2D coordinate charts that parameterize the body surface; its points are denoted $\boldsymbol{u}$. We assume that we have an invertible map, $\pi : \boldsymbol{u} \mapsto \boldsymbol{X}$, between points in the atlas and points on the surface. Intuitively, the atlas corresponds to the familiar texture space used in graphics, generalized to shapes of arbitrary topology. π is called the *parameterization*. π^{-1} is called the *coordinate map* and corresponds to the familiar texture map.

Table 1: *Vertex values and their interpretation*

$\boldsymbol{x}$	current position of body & skin	external input
$\boldsymbol{X}$	body position at $t = 0$	constant
$\mathbb{X}$	skin position at zero stress	computed by simulation
$\boldsymbol{u}$	body coordinate at $t = 0$	constant
$\mathbb{u}$	skin coordinate at zero stress	computed by simulation

The skin shares the shape of the body but is made of distinct material that can slide relative to it. Therefore, in parallel to the body, we represent the skin using a distinct 3D *skin space* in which the skin surface is embedded in a stress-free state[1] and its 2D *skin atlas*. Points in these spaces are denoted $\mathbb{X}$ and $\mathbb{u}$, respectively. Because the skin and body have the same geometry, we can use the same map π to parameterize them.

Keep in mind that the skin and body spaces are at "rest" in two different interpretations of that word, each appropriate for their purpose. The skin space is stress-free; the body space is at its initial strain. The skin space is used to measure the elastic deformation and hence stresses in the skin; the body space is used to measure the change in shape of the body during the animation. This formulation allows the skin to have an initial strain at $t = 0$, given by the initial vertex values $\mathbb{u}$.

It is important to remember that a mesh is a topological data structure, and any geometry is due to the values associated with the vertices of the mesh. For example, the map π^{-1} induces an equivalent triangulation of the body atlas. We can therefore think, equally well, that the mesh exists in any of the spaces shown. A single mesh triangle, and its counterparts in the spaces, are shown in Fig. 2.

3.2 Kinematics and Discretization

The motion of the skin is defined by two maps. The body's movement in space is specified by the *body motion* $\phi : \boldsymbol{X} \mapsto \boldsymbol{x}$. This is an arbitrary motion and deformation of the mesh, either specified by an artist or generated by another animation system. It is an input to our system. This map is given by its nodal values, $\boldsymbol{x}_i$ at vertex i, and linearly interpolated in between to produce a piecewise linear function ϕ, as is standard in finite element analysis. The maps π and π^{-1} are similarly represented after discretization by the nodal values $\boldsymbol{X}_i$ and $\boldsymbol{u}_i$. As we noted before, the same π can be used for the mapping between $\boldsymbol{X}, \boldsymbol{u}$ and $\mathbb{X}, \mathbb{u}$. Therefore, we precompute π using $\boldsymbol{X}$ and $\boldsymbol{u}$, and then we can use, for example, barycentric interpolation to look up $\mathbb{X}$ with $\mathbb{u}$.

The skin's movement relative to the body is accounted for by the *skin motion*, $\psi : \boldsymbol{X} \mapsto \mathbb{X}$, which is computed by our simulation. We represent it indirectly, using the *skin map*, $\kappa : \boldsymbol{X} \mapsto \mathbb{u}$; then $\psi = \pi \circ \kappa$. The reason we go through $\mathbb{u}$ is that it allows us to timestep the motion of the skin in 2D coordinates, rather than in 3D, thereby avoiding all the difficulties of constraining the skin to lie on the body surface. The skin map is discretized using nodal values, $\mathbb{u}_i$, and interpolated to produce a piecewise linear κ, as before.

With a slight abuse of notation for the sake of reducing clutter, we denote an array of stacked points, e.g., $(\mathbb{u}_0, \ldots, \mathbb{u}_n)^T$, using the same symbol we use for points, e.g., $\mathbb{u}$. The intent is clear from the context.

Table 1 summarizes the values stored at each mesh vertex.

[1]This is the most common scenario which we focus on, but some materials may not be embeddable in a stress-free state, particularly after plastic deformation. For extensions to handle such cases see, for example, [Wicke et al. 2010; Bargteil et al. 2007].

3.3 Jacobians

All our maps are piecewise linear and the Jacobian matrices of these maps are constant on each triangle. We can compute them using vertex values in the domain and range of the map, using the elegant construction of [Teran et al. 2003]. To be concrete, this procedure is illustrated using the Jacobian F of the composite map $\phi \circ \psi^{-1}$; this is also the deformation gradient of the skin that we will need later. But it must be emphasized that *all* the Jacobians can be computed in this manner, and we do so frequently in the rest of the paper.

For each triangle with vertex positions $\boldsymbol{x}_i, i = 0, 1, 2$, the edge vectors $\boldsymbol{d}_i = \boldsymbol{x}_i - \boldsymbol{x}_0, i = 1, 2$, are constructed. They are assembled into a matrix $D_{\boldsymbol{x}}$, with columns $\boldsymbol{d}_i$. $D_{\mathbb{X}}$ can be constructed similarly. By definition, $F = \partial \boldsymbol{x}/\partial \mathbb{X}$, so the columns of $D_{\mathbb{X}}$ are transformed as

$$F D_{\mathbb{X}} = D_{\boldsymbol{x}}. \tag{1}$$

If the points $\mathbb{X}_i$ have only 2 coordinates $D_{\mathbb{X}}$ is square, and we can compute $F = D_{\boldsymbol{x}} D_{\mathbb{X}}^{-1}$ (the more general case is addressed in the next section). Note that we did not have to explicitly use $\boldsymbol{X}$ values in the intermediate space or to construct Jacobians of π and ϕ. D also has a satisfying interpretation, as a discrete differential in a polyhedral space.

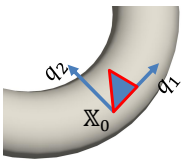
3.4 Elastic Forces

We represent the configuration of the skin and simulate its dynamics using $\mathbb{u}$. It is therefore tempting to also compute the elastic forces and the deformation gradient in the skin atlas. However, this is not ideal. First, since we represent shapes of arbitrary topology using a collection of charts, vertices of a skin triangle may lie in different charts, making triangle computations difficult. Second, material properties of the skin are difficult to specify in terms of $\mathbb{u}$ due to metric distortions caused by π^{-1}, which could be any coordinate map; it is more natural to specify these properties on the rest shape of the skin, in terms of $\mathbb{X}$. For these reasons, we compute the elastic forces in skin space, and transform them to the skin atlas using the (transposed) Jacobian in Eq. (4).

The deformation gradient of the skin, F, has been introduced in §3.3; since a mesh triangle in skin space is stress-free by definition, it can be used to compute the triangle's Green strain. However, computing this value is slightly complicated, since $\mathbb{X}$ has coordinates in the 3D embedding space, so $D_{\mathbb{X}}$ is a 3×2 matrix. Instead, we compute a reduced 3×2 deformation gradient $\bar{F}$, which completely captures the essential deformation of the triangle.

Let $D_{\mathbb{X}} = Q_{\mathbb{X}} R_{\mathbb{X}}$ be the thin QR decomposition of $D_{\mathbb{X}}$. Then $\bar{F} \stackrel{\text{def}}{=} F Q_{\mathbb{X}}$ and Eq. (1) can be rewritten as $F D_{\mathbb{X}} = F Q_{\mathbb{X}} R_{\mathbb{X}} = \bar{F} R_{\mathbb{X}} = D_{\boldsymbol{x}}$. Therefore the reduced deformation gradient is

$$\bar{F} = D_{\boldsymbol{x}} R_{\mathbb{X}}^{-1}. \tag{2}$$



Intuitively, the columns of $Q_{\mathbb{X}}$ form the axes of an orthonormal frame for the plane of the triangle, with origin at $\mathbb{X}_0$. The point with coordinates in this new frame is denoted $\bar{\mathbb{X}}$. In skin space it has 3D coordinates $\mathbb{X} = \mathbb{X}_0 + Q_{\mathbb{X}} \bar{\mathbb{X}}$. The columns of $R_{\mathbb{X}}$ are the edge vectors of the triangle in the new frame, and $R_{\mathbb{X}}$ is the 2×2 reduced differential in this frame.

$\bar{F}$ is used to compute the reduced Green strain

$$\bar{E} = \frac{1}{2}(\bar{F}^T \bar{F} - I) = \frac{1}{2}(Q_{\mathbb{X}}^T F^T F Q_{\mathbb{X}} - I) = Q_{\mathbb{X}}^T E Q_{\mathbb{X}}. \tag{3}$$

Thus $\bar{E}$ has the same non-zero eigenvalues as E, but simply ignores its irrelevant null space.

We assume that skin is a hyperelastic membrane, with strain energy function $W(\bar{E})$. We compute W symbolically as a function of the reduced vertex coordinates $\bar{\mathbb{X}}_i$, and derive the forces acting at vertex i of the triangle by symbolic differentiation using Maple™[Monagan et al. 2005]. Other derivatives required for implicit integration, see §3.5, are also computed symbolically. This is quite efficient, as shown in Section 4, since these computations are performed per triangle, with small matrices. We initially considered optimizing this computation based on additional physical insights, but found it was not worthwhile since it takes a small fraction of the total simulation time. The procedure is also very general, and could be used for any hyperelastic material model that can be expressed algebraically in terms of the reduced deformation gradient.

For example, for St. Venant-Kirchhoff material, the strain energy of a triangle is given by $W(\bar{E}) = A_{\mathbb{X}} \left(\frac{\lambda}{2}(\mathrm{tr}\bar{E})^2 + \mu \mathrm{tr}(\bar{E}^2)\right)$ where $A_{\mathbb{X}}$ is the area of the triangle in skin space (equal to $\frac{1}{2}\det R_{\mathbb{X}}$). λ and μ are the Lamé constants. Other types of materials can be used if desired [Volino et al. 2009].

The contribution of triangle j to the total force at vertex i is $\bar{f}_{ij} = -\partial W_j / \partial \bar{\mathbb{X}}_i$. We evaluate this quantity symbolically using Maple™. By the principle of virtual work, this force is transformed to skin space by $Q_{\mathbb{X}j}$ and then to the body atlas using the transpose of the triangle's Jacobian

$$\partial \mathbb{X} / \partial \boldsymbol{u}^T = (D_{\mathbb{X}} D_{\boldsymbol{u}}^{-1})^T = D_{\boldsymbol{u}}^{-T} D_{\mathbb{X}}^T = D_{\boldsymbol{u}}^{-T} R_{\mathbb{X}}^T Q_{\mathbb{X}}^T. \quad (4)$$

Here the triangle index is suppressed for clarity, and the thin QR decomposition of $D_{\mathbb{X}}$ is substituted. The transformed force in the body atlas simplifies to

$$\boldsymbol{f}_{ij} = D_{\boldsymbol{u}j}^{-T} R_{\mathbb{X}j}^T Q_{\mathbb{X}j}^T Q_{\mathbb{X}j} \bar{f}_{ij} = D_{\boldsymbol{u}j}^{-T} R_{\mathbb{X}j}^T \bar{f}_{ij}. \quad (5)$$

The total vertex force $\boldsymbol{f}_i$ is the sum of the contributions $\boldsymbol{f}_{ij}$ from incident faces j.

3.5 Dynamics

The configuration of the skin is completely determined by the vertex values $\mathbb{u}$, and the external input $\boldsymbol{x}$. Therefore the space $\mathbb{U}$ of all $\mathbb{u}$ values at mesh vertices is a skin *configuration space*, and the standard Lagrangian procedure can be used to write the dynamics of the skin entirely in terms of the stacked vector $\mathbb{u}$ and its derivatives. Motion of the entire skin is equivalent to the motion of a point in $\mathbb{U}$. The high dimensional space $\mathbb{U}$ should not be confused with the 3D Euclidean space $\mathbb{X}$ in which the skin is embedded.

However, there is still the important choice to make whether the value at mesh vertex i, $\mathbb{u}_i$, is the coordinate of a fixed point on the skin (Lagrangian discretization) or of the skin material that is currently at a fixed point on the body (Eulerian discretization). In the Lagrangian setting the mesh is fixed in the skin and moves on the body during simulation; in the Eulerian setting the mesh is fixed on the body and moves on the skin. We choose an Eulerian discretization for the reasons discussed in §1.

The generalized inertia (or mass) matrix M is computed as follows. Let $\mathbb{v}$ be the material velocity of the skin point $\mathbb{u}$ (which is at body location $\boldsymbol{u}$ in the Eulerian setting). Its velocity in space is then $\dot{\boldsymbol{x}} = \Gamma \mathbb{v}$ where $\Gamma \stackrel{\mathrm{def}}{=} \partial \boldsymbol{x} / \partial \boldsymbol{u}$ (see Fig. 2). Note that the Jacobian from the body atlas is needed here; the skin has no material velocity in the skin atlas. Then, the kinetic energy of the skin is

$$T = \frac{1}{2} \int_{A_{\boldsymbol{u}}} \rho \mathbb{v}^T \Gamma^T \Gamma \mathbb{v} \, dA_{\boldsymbol{u}} = \frac{1}{2} \mathbb{v}^T \left(\int_{A_{\boldsymbol{u}}} \rho \Gamma^T \Gamma \, dA_{\boldsymbol{u}} \right) \mathbb{v}$$

The matrix in parentheses is the generalized inertia M. ρ is areal density and A is area.

The Jacobian $\Gamma = D_{\boldsymbol{x}} D_{\boldsymbol{u}}^{-1}$ is constant within each triangle. Computing it is straightforward; $D_{\boldsymbol{u}}^{-1}$ is also constant in time (and can be precomputed). We approximate the velocity to be constant in each triangle j, with $\mathbb{v}_j = \frac{1}{3} \sum_{i \in vert(j)} \mathbb{v}_i$. Then the inertia contribution of triangle j is

$$M_j = \mathbb{m}_j \Gamma_j^T \Gamma_j. \quad (6)$$

Here $\mathbb{m}_j$ is the mass of the skin in triangle j; this scalar value is easy to compute using the density and vertex values in skin space (already available from the elasticity computation in §3.4).

Vertex inertias are computed by adding $1/3$ the inertia of each incident triangle. These 2×2 vertex inertias are assembled into the global block-diagonal inertia matrix M.

After time discretization using the linearly implicit method widely used in graphics [Baraff and Witkin 1998], we get

$$\left(M + h^2 \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \right) \tilde{\mathbb{v}}^{(k+1)} = M \mathbb{v}^{(k)} + h(\boldsymbol{f}^{(k)} + \boldsymbol{b}^{(k)}). \quad (7)$$

We use the tilde in $\tilde{\mathbb{v}}^{(k+1)}$ to remind ourselves that this is the material velocity of the point that *was* at a mesh vertex at step k, and we will need an advection step to arrive at the final Eulerian velocity $\mathbb{v}^{(k+1)}$. Also, h is the size of the time step; $\boldsymbol{f}$ is the elastic force (§3.4); $\boldsymbol{b}$ is body force due to gravity or other phenomena (including the user), applied in the physical space and transformed to the body atlas by Γ^T. To obtain $\partial \boldsymbol{f} / \partial \boldsymbol{u}$, we evaluate $\partial \bar{f} / \partial \bar{\mathbb{X}}$ symbolically using Maple™, and then rotate and transform it in a similar manner to Eq. (5). Note that M depends on the skin configuration, which gives rise to a "quadratic velocity vector" term in the dynamics as well. This is not significant for typical skin movements, so we ignore it. The integration scheme is stable, but adds time-step dependent numerical damping, which is not a significant problem for skin movement. If desired, other types of integration schemes can be used without modifying the algorithm.

Next, the velocity field is advected using the stable semi-Lagrangian method [Stam 1999; Feldman et al. 2005]. More sophisticated methods developed in fluid mechanics could be used if needed [Kim et al. 2005; Selle et al. 2008; Lentine et al. 2011]. We are agnostic to the particular choice, and assume that an adequate advection routine is available such that for any material quantity q,

$$q = \mathtt{advect}(\mathbb{v}, h, \tilde{q}). \quad (8)$$

Using this method we advect the velocity, to obtain $\mathbb{v}^{(k+1)}$. Finally, the velocity is integrated by advecting skin positions.

$$\mathbb{u}^{(k+1)} = \mathtt{advect}(\mathbb{v}, h, \tilde{\mathbb{u}}^{(k+1)}). \quad (9)$$

3.6 Coupling Skin and Body

The motion of the body influences the motion of the skin in complex ways. This includes non-penetration constraints and viscous resistance to sliding. In animal skin the connective tissue fibers anchoring skin to subcutaneous structures have highly nonlinear elastic behavior; the stress-strain curve has a low force "toe" region in the physiological range, but becomes very stiff beyond that region. Our goal is not to model these biological tissues in detail, but to capture some of their essential features in an efficient simulation. In addition, we would like to provide sufficient modeling flexibility

so that an artist can choose to have elastodynamic skin only on a region of interest of the body, and have a small number of parameters to control the behavior of skin relative to the body.

Constraints provide a very general method for attaching skin to the body and for setting boundary conditions on the region of interest to be simulated. Attachments at vertex positions are particularly easy to enforce in the Eulerian setting, since the constraints are collocated with state variables, as observed by [Sueda et al. 2011]. At the velocity level, the constraints on vertex i are of the general form $G\mathbb{v}_i = g$. Skin velocity is obtained by solving the KKT equations for the constrained dynamical system

$$\begin{pmatrix} M^* & G^T \\ G & 0 \end{pmatrix} \begin{pmatrix} \tilde{\mathbb{v}}^{(k+1)} \\ \lambda \end{pmatrix} = \begin{pmatrix} f^* \\ g \end{pmatrix}, \tag{10}$$

where λ is the vector of Lagrange multipliers, $M^* = M + h^2\partial \boldsymbol{f}/\partial \boldsymbol{u}$, and $f^* = M\mathbb{v}^{(k)} + h(\boldsymbol{f}^{(k)} + \boldsymbol{b}^{(k)})$ (compare Eq. (7)).

This method can model a variety of important conditions. When the skin is fixed to the body at vertex location $\boldsymbol{u}_i$, the velocity $\mathbb{v}_i = 0$. This is enforced by setting $G = I$ and $g = 0$. More generally, we can constrain a skin vertex to not move along the normal $\boldsymbol{a}$ to a constraint curve, but allow it to slide along the curve. In this case, the constraint is $\boldsymbol{a}^T\mathbb{v}_i = 0$ and is enforced by setting $G = \boldsymbol{a}^T$ and $g = 0$.

Body movement. The main input to the simulation is the (arbitrary) deformation of the body mesh, $\boldsymbol{x}^{*(k+1)}$ at the new time step $k+1$, provided by the user. Skin points that are not explicitly constrained as above are still influenced by implicit contact constraints. Therefore we consider $\boldsymbol{x}^{*(k+1)}$ a target vertex displacement

$$\Delta\boldsymbol{x} = \boldsymbol{x}^{*(k+1)} - \boldsymbol{x}^{(k)}, \tag{11}$$

which is modified to a feasible displacement $\Delta\boldsymbol{x}'$ that enforces these constraints approximately in each time step.

$\Delta\boldsymbol{x}$ is decomposed into a normal component $\Delta\boldsymbol{x}_n$ and a tangential component $\Delta\boldsymbol{x}_t$. To enforce non-penetration and non-separation constraints, $\Delta\boldsymbol{x}_n$ is left unchanged, so that the skin tracks the body in the normal direction. In the tangential direction, there is some skin damping and also a limit on the amount of strain ($\Delta_{\max}$) that models the biphasic behavior of collagen in skin. We obtain a corrected displacement $\Delta\bar{\boldsymbol{x}}_t = \frac{\Delta\boldsymbol{x}_t}{||\Delta\boldsymbol{x}_t||}\min(||\Delta\boldsymbol{x}_t||, \Delta_{\max})$ and scale it with a parameter ζ to get

$$\Delta\boldsymbol{x}' = \Delta\boldsymbol{x}_n + \zeta\Delta\bar{\boldsymbol{x}}_t \tag{12}$$

ζ is a user specified "inverse damping" parameter: $\zeta = 1$ approximates infinite friction so that the skin material sticks tightly on the body mesh; $\zeta = 0$ makes the sliding motion highly underdamped (there is still resistance to motion due to tension from neighboring vertices and artificial damping from implicit integration). $\boldsymbol{x}$ in time step k is updated with $\Delta\boldsymbol{x}'$ instead and transformed to skin using the pseudoinverse:

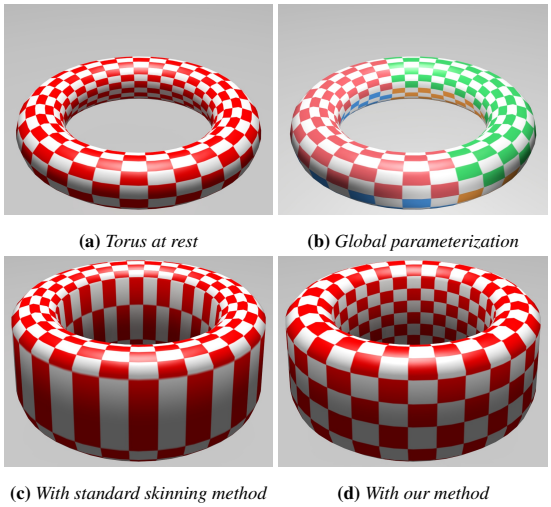
$$\Delta\mathbb{u} = \Gamma^\dagger\Delta\boldsymbol{x}'. \tag{13}$$

Note that since we compare the target shape to the actual skin shape at each time step in Eq. (11), all errors are eventually corrected and there is no constraint drift.

The pseudocode for our implementation showing all the steps and equations are shown in Alg. 1.

Algorithm 1 Thin Skin Elastodynamics

```
// Initialization
Build a discretization of π with X and u
// Simulation loop
while simulating do
    Move mesh vertices to x^{*(k+1)} via external driver
    // Dynamics coupling
    for all verts i do
        Get feasible displacement Δx'_i                    // Eq. (12)
        Update: x_i^{(k+1)} = x_i^{(k)} + Δx'_i
    end for
    for all triangles j with verts i do,
        Compute Δu_ij                                      // Eq. (13)
    end for
    Advect with Δu to obtain ũ^{(k+1)}                     // Eq. (8)
    // Elastic force
    for all verts i do
        Look up X_i with u_i via π
    end for
    for all triangles j with verts i do
        Compute elastic force f_ij                         // Eq. (5)
        Compute M_j                                        // Eq. (6)
    end for
    // Time integration
    Form KKT system and solve for ṽ^{(k+1)}                // Eq. (10)
    Advect the velocity ṽ^{(k+1)} to obtain v^{(k+1)}      // Eq. (8)
    Advect skin positions ũ^{(k+1)} to obtain u^{(k+1)}    // Eq. (9)
end while
```



(a) *Torus at rest* **(b)** *Global parameterization*

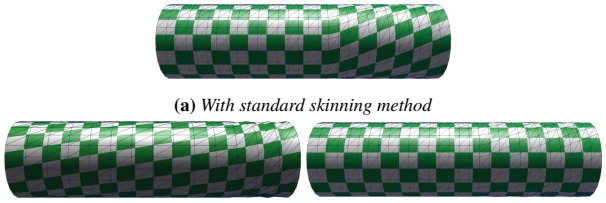
(c) *With standard skinning method* **(d)** *With our method*

Figure 3: *Torus Deformation*

4 Results

We implemented our system in C++, and ran the simulations on a 2.66 GHz Intel Core i5 computer with 4 GB of RAM. The code is single-threaded and uses csparse [Davis 2006] for solving the sparse linear system. We use 1 millisecond time steps for all the demos.

Torus deformation. Fig. 3 shows our simulation result on a closed surface. The torus is parameterized by four adjacent charts as shown in Fig. 3b. Using traditional skinning, non-uniform deformations can cause large, undesirable distortion of the texture, as shown in Fig. 3c. With our method the surface material slides to uniformly distribute the distortion over the entire surface, as shown in Fig. 3d.



(a) *With standard skinning method*

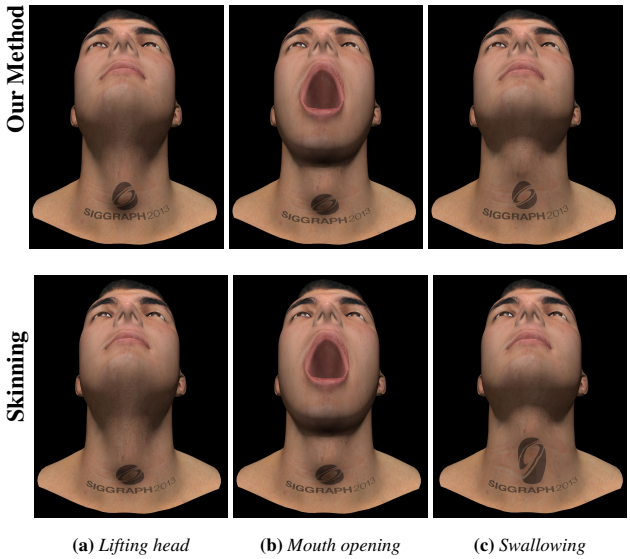
(b) *Fixed boundary constraints on both sides*

(c) *Fixed boundary constraints on the left, sliding constraints on the right*

Figure 4: *Cylinder twisting*

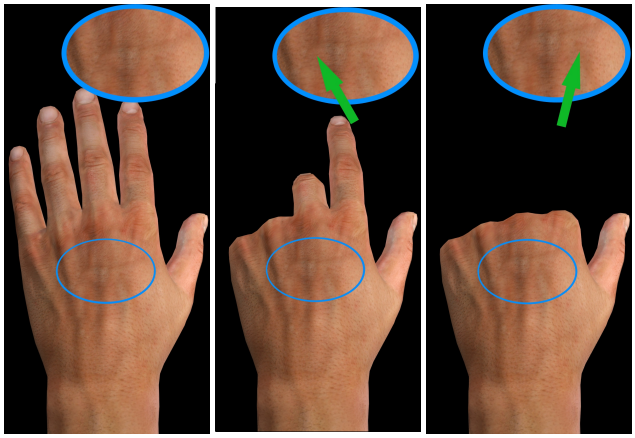
Cylinder twisting. The cylinder example in Fig. 4 shows the ability of our method to deal with different boundary constraints. If only the right half of the cylinder is twisted, traditional skinning produces deformation only on the right hand side of the cylinder. Using our method, the deformation of the skin can be naturally propagated into the left half of the cylinder depending on the boundary constraint assigned by user. If the right boundary of the skin is fixed to the body mesh, our method produces a uniform twist of the entire skin (Fig. 4b). When the right boundary is allowed to slide along the body mesh, the body mesh slides independently under the skin (Fig. 4c).

Head movement. Our method can simulate realistic skin deformations of various body parts. We first present realistic animation of the head and neck including the Adam's apple (laryngeal prominence) as a convincing example. Since the Adam's apple slides under the skin, it produces visually interesting deformations of the neck: when the head is lifted and the mouth is opened, the skin slides along the neck while the Adam's apple does not move much. On the other hand, when we swallow, the Adam's apple moves up and down while the skin remains relatively static. Although such deformation behaviors are critical for visually realistic head animation, they cannot be simulated using traditional skinning methods. Using our method, we can directly simulate these realistic behaviors. Fig. 5 compares the results of (a) head lifting, (b) mouth opening, and (c) swallowing, using our method (top row) and traditional skinning (bottom row). We draw a tattoo on the character's neck to highlight the skin deformation. As expected, our method can simulate realistic skin deformations described above, while the traditional skinning produces opposite results; no skin deformations during head lifting and mouth opening, and large skin distortion during swallowing.



(a) *Lifting head* **(b)** *Mouth opening* **(c)** *Swallowing*

Figure 5: *Head movement: the three figures in the top row show the results of our proposed method, and the bottom three figures are obtained using the standard skinning method.*



(a) **(b)** **(c)**

Figure 6: *The skin slides on the back of the hand in response to flexion and extension of the fingers. In this example, our method moves the normal map along with the standard texture map.*

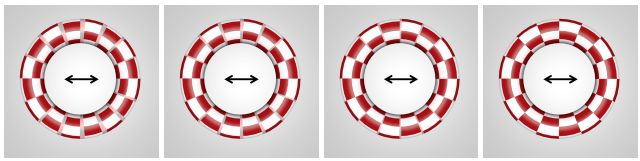
Hand flexing. Our method is not limited to the deformation of texture pixels; it can be used with other sources such as normal maps. In the hand example shown in Fig. 6, the wrinkles and veins are represented in a normal map. When we flex the fingers sequentially from pinky to index finger, the wrinkles and veins of the dorsal side first slide toward the pinky (direction of the arrow in Fig. 6b) and then move toward the index finger (direction of the arrow in Fig. 6c).

Tight clothes sliding on torso. Our method can also be used to animate tight-fitting clothes worn by a character. Without separately simulating the deformation of a thin shell on skin, our method can generate realistic cloth deformations by simply making the cloth material slide freely along the body mesh. Fig. 7 compares the results of our method to standard skinning for upper body animation. When the character shrugs or lifts his left arm, the whole cloth is naturally stretched in our method (Figs. 7b and 7c top), while in the traditional method only the left part of the cloth is stretched for arm lifting (Fig. 7b bottom) and only the shoulder part is stretched for shrugging (Fig. 7c bottom). Since we are not introducing additional simulation steps for clothing, the deformation can be computed very efficiently: with 3320 DoFs and 278 constraints, the whole computation took 164.04ms per frame.

Torus shaking. Since different types of skin interact with the underlying body in different ways, our method gives the animator freedom to choose the dynamic coupling of skin to body by simply adjusting the ζ variable defined in §3.6. If $\zeta = 1$, the skin rigidly sticks to the body mesh and does not produce any sliding motion. If $\zeta = 0$, the sliding behavior of skin is heavily affected by the motion of the body mesh. Fig. 8 shows an example of a torus shaken by sinusoidal motion. When $\zeta = 0$, our method can simulate the visually realistic dynamic effect of skin material that lags

(a) *Rest pose* **(b)** *Lifting arm* **(c)** *Shrug*

Figure 7: *Tight clothes slide on the torso during movement. The upper figures are simulated with our method while the bottom figures are obtained using standard skinning.*



(a) $\zeta = 0$ **(b)** $\zeta = 0.3$ **(c)** $\zeta = 0.6$ **(d)** $\zeta = 1$

Figure 8: *Torus shaking: the skin slides on the surface when a horizontal rigid motion is applied to the torus. ζ values control the influence of body movement. Skin motion is visualized as motion blur; the video shows this more clearly.*

and overshoots the motion of the body mesh.

Dinosaur walking. The dynamic coupling described above can be efficiently used in character animation to produce a visually realistic jiggling of skin material. The dinosaur example provided in the video shows that the skin jiggles when the dinosaur suddenly stops moving.

4.1 Performance

All the examples illustrated in this paper are simulated at interactive rates with our method. Table 2 tabulates performance statistics for above examples. In almost all of our examples, the linear solver dominates the overall computation time.

5 Limitations and Future Work

Our method has several limitations. To make it easy to integrate with any animation pipeline, we chose one-way coupling between body and mesh. Two-way coupling is useful when the skin (or a tight fitting suit) is to influence the body's dynamics. Most biological tissues are incompressible but we did not enforce this in our model. This is in part due to the fact that wrinkled skin is not incompressible at a coarse scale due to subgrid folds that can change surface area. It also removes the restriction that the input body mesh deformation is also area preserving. Incompressibility constraints could be added at the velocity level if essential for some applications, as they are done in fluids simulation. Simulating or synthesizing wrinkles and buckling from the computed stresses [Rohmer et al. 2010] could be included in this framework, but were not. Inversions could occur in some cases, and so an approach similar to Irving et al. [2004] would be useful. We assumed that the skin can be embedded in a stress-free state on the body surface at some configuration, although this may not be possible for some biological materials. It should be possible to account for this using the same methods used for representing plastically deformed solids. Even though our framework is general, our examples used a simple St. Venant-Kirchhoff material. We tried a more biomechanically realistic Fung-like skin model with an exponential term to prevent large deformation, but it introduced numerical instabilities and may need new types of integrators. We also did not model the cutaneous ligament, which binds the skin to the underlying tissues. This is an area for future work. Currently we only use low order methods for discretizing space and time, following the practice in animation, which favors efficiency over accuracy of the simulation.

Table 2: *Statistics of examples. Cons = Constraints, T = Total time per frame in ms, T_{LS} = Time for linear solver in ms, T_R = Time for the rest of steps in ms. "torus" = torus deformation, "torus*" = torus shaking.*

Example	Tris	DoFs	Cons	T	T_{LS}	T_R
torus	1152	1152	0	22.44	14.73	7.71
cylinder	1276	1320	44	33.72	25.67	8.05
head	1593	1724	258	50.75	39.95	10.80
hand	445	516	118	9.00	5.83	3.17
cloth	3162	3320	278	164.04	144.75	19.29
torus*	1152	1152	0	23.37	14.45	8.92
dinosaur	461	560	198	6.27	2.73	3.54

6 Conclusions

We have introduced a new method for simulating human-like skin that is in close contact with the underlying structures of the body. The key to the method is a novel Eulerian discretization of thin membranes that are constrained to slide on surfaces of arbitrary topology. This discretization makes the simulation very robust since the major problems of dealing with contact between the skin and the body are avoided. The method is also easy to implement and efficient. Since it is simple to integrate the method with any animation pipeline as a post-process, we hope it will be widely useful.

Acknowledgment The authors would like to thank David Levin and the anonymous reviewers for their help. This work was funded in part by grants from the Canada Research Chairs Program, NSERC, Peter Wall Institute for Advanced Studies, and CFI.

References

ALBRECHT, I., HABER, J., AND SEIDEL, H.-P. 2003. Construction and animation of anatomically based human hand models. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 98–109.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proc. SIGGRAPH 1998*, Annual Conference Series, 43–54.

BARGTEIL, A. W., WOJTAN, C., HODGINS, J. K., AND TURK, G. 2007. A Finite Element Method for Animating Large Viscoplastic Flow. *ACM Trans. Graph. 26*, 3 (July), 16:1–16:8.

BEELER, T., HAHN, F., BRADLEY, D., BICKEL, B., BEARDSLEY, P., GOTSMAN, C., SUMNER, R., AND GROSS, M. 2011. High-quality passive facial performance capture using anchor frames. *ACM Trans. Graph. 30*, 4 (July), 75:1–75:10.

CHOE, B., LEE, H., AND KO, H. 2001. Performance-driven muscle-based facial animation. *The Journal of Visualization and Computer Animation 12*, 2, 67–79.

DAVIS, T. A. 2006. *Direct Methods for Sparse Linear Systems*. SIAM Book Series on the Fundamentals of Algorithms. SIAM.

DONG, S., BREMER, P.-T., GARLAND, M., PASCUCCI, V., AND HART, J. C. 2006. Spectral surface quadrangulation. *ACM Trans. Graph. 25*, 3 (July), 1057–1066.

FAN, Y., LEVIN, D. I. W., LITVEN, J., AND PAI, D. K., 2013. Eulerian-on-Lagrangian simulation. To appear in ACM Transactions on Graphics.

FELDMAN, B. E., O'BRIEN, J. F., KLINGNER, B. M., AND GOKTEKIN, T. G. 2005. Fluids in deforming meshes. In *ACM SIGGRAPH/Eurographics symposium on Computer Animation*, 255–259.

GOURRET, J.-P., THALMANN, N. M., AND THALMANN, D. 1989. Simulation of object and human skin formations in a grasping task. In *Computer Graphics*, vol. 23, 21–30.

GRINSPUN, E., HIRANI, A. N., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete shells. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 62–67.

HUANG, H., ZHAO, L., YIN, K., QI, Y., YU, Y., AND TONG, X. 2011. Controllable hand deformation from sparse examples with rich details. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 73–82.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *ACM SIGGRAPH/Eurographics symposium on Computer Animation*, 131–140.

JAMES, D. L., AND TWIGG, C. D. 2005. Skinning mesh animations. *ACM Trans. Graph. 24*, 3 (July), 399–407.

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

KIM, B., LIU, Y., LLAMAS, I., AND ROSSIGNAC, J. 2005. Flowfixer: using BFECC for fluid simulation. In *Proceedings of the First Eurographics conference on Natural Phenomena*, 51–56.

KRY, P. G., JAMES, D. L., AND PAI, D. K. 2002. Eigenskin: real time large deformation character skinning in hardware. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 153–159.

LEE, S.-H., SIFAKIS, E., AND TERZOPOULOS, D. 2009. Comprehensive biomechanical modeling and simulation of the upper body. *ACM Trans. Graph. 28*, 4 (Sep), 99:1–99:17.

LENTINE, M., AANJANEYA, M., AND FEDKIW, R. 2011. Mass and momentum conservation for fluid simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 91–100.

LEVIN, D. I. W., LITVEN, J., JONES, G. L., SUEDA, S., AND PAI, D. K. 2011. Eulerian solid simulation with contact. *ACM Trans. Graph. 30*, 4 (July), 36:1–36:9.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proc. SIGGRAPH 2000*, Annual Conference Series, 165–172.

MAGNENAT-THALMANN, N., LAPERRIRE, R., THALMANN, D., AND MONTRAL, U. D. 1988. Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interface 88*, 26–33.

MAILLOT, J., YAHIA, H., AND VERROUST, A. 1993. Interactive texture mapping. In *Proc. SIGGRAPH 1993*, Annual Conference Series, 27–34.

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

MOHR, A., AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. *ACM Trans. Graph. 22*, 3 (July), 562–568.

MONAGAN, M. B., GEDDES, K. O., HEAL, K. M., LABAHN, G., VORKOETTER, S. M., MCCARRON, J., AND DEMARCO, P. 2005. *Maple 10 Programming Guide*. Maplesoft, Waterloo ON, Canada.

PARK, S. I., AND HODGINS, J. K. 2006. Capturing and animating skin deformation in human motion. *ACM Trans. Graph. 25*, 3 (July), 881–889.

QIN, H., AND TERZOPOULOS, D. 1996. D-NURBS: A Physics-Based Framework for Geometric Design. *IEEE Transactions on Visualization and Computer Graphics 2*, 1, 85–96.

ROHMER, D., POPA, T., CANI, M.-P., HAHMANN, S., AND SHEFFER, A. 2010. Animation wrinkling: augmenting coarse cloth simulations with realistic-looking wrinkles. *ACM Trans. Graph. 29*, 6 (Dec.), 157:1–157:8.

SELLE, A., FEDKIW, R., KIM, B., LIU, Y., AND ROSSIGNAC, J. 2008. An unconditionally stable maccormack method. *J. Sci. Comput. 35*, 2-3 (June), 350–371.

SHEFFER, A., PRAUN, E., AND ROSE, K. 2006. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis. 2*, 2 (Jan.), 105–171.

SIFAKIS, E., NEVEROV, I., AND FEDKIW, R. 2005. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph. 24*, 3 (July), 417–425.

SIFAKIS, E., HELLRUNG, J., TERAN, J., OLIKER, A., AND CUTTING, C. 2009. Local flaps: A real-time finite element based solution to the plastic surgery defect puzzle. *Studies in Health Technology and Informatics 142*, 313–8.

STAM, J. 1999. Stable fluids. In *Proc. SIGGRAPH 1999*, Annual Conference Series, 121–128.

STAM, J. 2003. Flows on surfaces of arbitrary topology. *ACM Trans. Graph. 22*, 3 (July), 724–731.

SUEDA, S., KAUFMAN, A., AND PAI, D. K. 2008. Musculotendon simulation for hand animation. *ACM Trans. Graph. 27*, 3 (Aug.), 83:1–83:8.

SUEDA, S., JONES, G. L., LEVIN, D. I. W., AND PAI, D. K. 2011. Large-scale dynamic simulation of highly constrained strands. *ACM Trans. Graph. 30*, 4 (July), 39:1–39:9.

TERAN, J., BLEMKER, S., HING, V. N. T., AND FEDKIW, R. 2003. Finite volume methods for the simulation of skeletal muscle. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 68–74.

TERZOPOULOS, D., AND WATERS, K. 1990. Physically-based facial modelling, analysis, and animation. *The Journal of Visualization and Computer Animation 1*, 2 (Dec.), 73–80.

VOLINO, P., MAGNENAT-THALMANN, N., AND FAURE, F. 2009. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Trans. Graph. 28*, 4 (Sept.), 105:1–105:16.

WICKE, M., RITCHIE, D., KLINGNER, B. M., BURKE, S., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2010. Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graph. 29* (July), 49:1–49:11.

WILHELMS, J., AND GELDER, A. V. 1997. Anatomically based modeling. In *Proc. SIGGRAPH 1997*, Annual Conference Series, 173–180.

WU, Y., KALRA, P., AND THALMANN, N. 1996. Simulation of static and dynamic wrinkles of skin. In *Computer Animation'96. Proceedings*, 90–97.

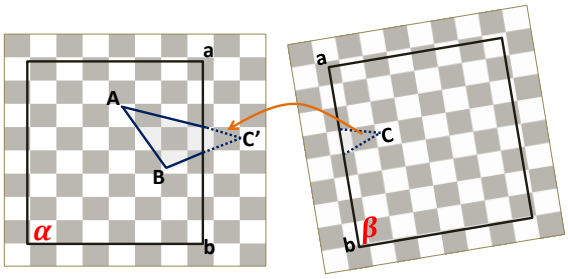


Figure 9: *A triangle spanning two charts. α and β are two adjacent patches. At initialization, we make sure that the texture extends beyond the border of each patch. Whenever a triangle transitions from one patch to another, we transit 1 or 2 vertices from the coordinate system of one patch to the coordinate system of another patch.*

A Appendix: Implementation Details

A.1 Parameterization

Our framework can use any base complex-based parameterization with a well-defined transition function between charts. For an overview of different parameterization choices, we refer the reader to the review article by Sheffer et al. [2006]. For our implementation, we chose to use the approach of Dong et al. [2006]: we build a globally smooth domain for simulation by parameterizing the surface as an atlas of rectangular charts (or base complex) with Laplacian operation. The transition functions include both translation and rotation between any two adjacent charts oriented in a right-hand coordinate system, as described by Stam [2003]. We then solve for the parameterization by solving a linear system of equations, where we assign normalized discrete harmonic weights to the vertices and incorporate the transitions between the adjacent charts.

A.2 Rendering

Although transition between charts do not pose any difficulties for the simulator, it is problematic when rendering textured triangles. If one or two vertices jump to another chart, but if other vertices remain in the original chart, then the triangle spanned by the three vertices contains a break in the texture map (Fig. 9). We fix this by extending the pixel region of each chart with the pixels from adjacent texture chart regions. When the triangle moves across the border, we transit its coordinates to make sure all three vertices are in the same chart. Because the global parameterization spans over adjacent charts, the overlapped border region is C^1 continuous. Therefore, no artifact will be introduced during the transition, and the spanned triangle can map to the correct pixel regions.