EUROGRAPHICS 2005 / M. Alexa and J. Marks
(Guest Editors)

Volume 24 (2005), Number 3 (to appear)

Efficient Animation of Point-Sampled Thin Shells

Martin Wicke Denis Steinemann Markus Gross

ETH Zurich

Abstract

We present a novel framework for the efficient simulation and animation of discrete thin shells. Our method takes a point sampled surface as input and performs all necessary computations without intermediate triangulation. We discretize the thin shell functional using so-called fibers. *Such fibers are locally embedded parametric curves crisscrossing individual point samples. In combination, they create a dense mesh representing the surface structure and connectivity for the shell computations. In particular, we utilize the fibers to approximate the differential surface operators of the thin shell functional. The polynomials underlying the fiber representation allow for a robust and fast simulation of thin shell behavior. Our method supports both elastic and plastic deformations as well as fracturing and tearing of the material. To compute surfaces with rich surface detail, we designed a multiresolution representation which maps a high-resolution surface onto a fiber network of lower resolution. This makes it possible to animate densely sampled models of very high surface complexity. While being tuned for point sampled objects, the presented framework is versatile and can also take triangle meshes or triangle soups as input.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism–Animation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling–Physically based modeling I.6.8 [Simulation and Modeling]: Types of Simulation–Animation

1. Introduction

The efficient simulation of physical material behavior has become a very important area of computer graphics. Its wide spectrum of applications ranges from the medical simulation to games and entertainment. Examples include the simulation of cloth [BFA02, BMF03], human tissue [KGC*96], fracturing materials [OH99], and others. In many cases, robustness, high visual accuracy, and speed are the predominant design criteria, whereas numerical precision or the correct reproduction of the underlying physics are considered less important. The design of such methods tailored towards Computer Graphics' needs has a long tradition. Since its inception by the milestone work of [TPBF87], various simulation techniques from mass-spring systems to finite element (FE) methods have been explored. FE methods for three-dimensional deformation physics are highly complex and require the solution of partial differential equations. In recent years, substantial improvements have been made with regard to material properties [GBO04] and computational efficiency [MG04].

A large class of interesting real world objects is thin-walled. When applied to very thin, almost two-dimensional objects, the models derived from three-dimensional continuum mechanics degenerate to thin shell functionals. Such thin shells are three-dimensional bodies with one geometric dimension significantly smaller than the other two. They have a non-planar rest shape, which distinguishes them from thin plates.

The fundamental theory underlying thin shells is based on the Kirchhoff theory of thin plates and the Kirchhoff Love theory of thin shells [GZ68]. The thin shell functional depends on surface curvature and, thus, contains second order derivatives of the displacement function. FE solutions for thin shells are more demanding than for conventional 3D continuum mechanics [Bat95]. The thin shell energy is typically broken down into a *stretching* (or membrane) term and a *bending* (or flexural) term. While the former depends on a first order differential operator, the latter one includes a second order term. In addition, the full formulation includes nonlinear geometric differentials, such as curvature, which are very often linearized to become numerically tractable.

In this paper we introduce a novel framework to efficiently

simulate thin shells on unstructured, point sampled input surfaces. Our contributions are summarized as follows:

Unlike earlier approaches [GHDS03, CG91] our method does not require a consistent triangulation of the underlying object, nor do we need a global surface parametrization. Instead, we introduce the concept of so-called *fibers*, a set of locally interpolating polynomial curves crisscrossing the point set. The entirety of fibers constitutes a dense network which represents the surface structure and the connectivity needed for simulation. The polynomial nature of the fibers allows us to conveniently discretize the first and second order terms of the thin shell functional per surface sample and in discrete angular directions. As we will show, the stretching component can be evaluated by sampling the changes in arc length of all fibers crossing through a sample. Likewise, the second order component is computed by averaging over the normal curvatures of all such fibers. Our framework lends itself well to the simulation of elasticity, plasticity, tearing, and fracture. In case of fracture, the point sampled representation permits efficient restructuring of the surface without the burden of maintaining a consistent triangulation. Our multiresolution setting combines a high resolution object with a coarsely sampled set of simulation nodes, enabling the simulation of very densely sampled input surfaces.

While the presented implementation is tailored towards point samples, our approach is versatile and works for a variety of surface representations including triangle meshes or polygon soup. We only assume a sufficiently dense sampling of the underlying surface.

2. Related Work

Thin shells have been introduced to Computer Graphics as early as 1987 in the framework of [TPBF87], who described the thin shell energy in terms of first and second order metric tensors. The efficient numerical solution of these functionals, however, remained a major challenge largely prohibiting the use of thin shells for animation. In contrast, plate based membranes have been used widely to simulate cloth dynamics in graphics. Examples from the rich body of literature include [BW98, BMF03], who present methods for cloth simulation. Others [BFA02, THMG04] employed mass spring systems for a fast approximation of elastic behaviour.

[CG91] proposed an FE based modeling system utilizing both stretching and bending energies for physics based surface deformation. To make the formulation computationally tractable, the first and second order tensors were linearized. Subsequently, [KGC*96] presented a framework for surgical simulation which employed a similar shell representation.

Later, [COS00] investigated the utility of subdivision surfaces to efficiently simulate thin shells. They demonstrated that the multiresolution nature of subdivision surfaces makes complex shell simulations more efficient. This method was adopted by [GTS02]. While being computationally elegant, local topological changes, such as occurring during fracture or tearing, pose a great challenge. The issue of adaptivity for graphics simulation has been addressed by the versatile simulation framework of [GKS02]. The authors presented an adaptive, hierarchical refinement method for FE simulation supporting 3D continuum mechanics and thin shells. Tearing and fracture, however, have not been addressed. Recently, [MBF04] presented a virtual node algorithm for the fast animation of fracturing materials with an emphasis on thin shells. This method retains a consistent connectivity upon the topological changes occurring during material fracture.

The research closest to ours is the discrete shell framework of [GHDS03]. The authors discretize both stretching and bending components of the thin shell functional on the underlying triangle mesh: In particular, bending energy is represented by a discrete curvature operator [MDSB02], and stretching energy is approximated by triangle edge lengths and areas. Compared to our work, these discretizations draw upon the triangle as a fundamental building block and thus require a consistent mesh at all times. Topological changes during fracture demand mesh restructuring and become difficult to handle.

Conversely, our discretizations are based on a one-dimensional, higher-order polynomial primitive, the so-called *fiber*. Our fiber network is generated from an unstructured point sampled input surface and locally connects adjacent points without requiring a globally consistent connectivity. Topological changes, as introduced by fracture or tearing, are handled gracefully.

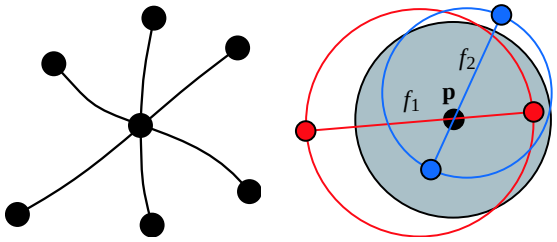
Our research was inspired by recent progress in the field of point based representations. Early work of Szelisky and Tonnesen [ST92] presented a framework for physics based surface modeling based on particle sampling. Recently, [PKKG03] presented methods to deform high resolution surfaces in real-time and [MKN*04] introduced a framework for meshless FE based on point sampled input geometry.

The remainder of the paper is organized as follows: In Section 3, we briefly review the mechanics of thin shells and derive our discretization of stretching and bending energies using fibers. Section 4 addresses the dynamic simulation of the discrete thin shell. In Sections 5 and 6 we discuss the creation of the fiber network from an unstructured input point model and present our multiresolution setting. Some more advanced issues, including plasticity and fracture, are discussed in Section 7. Finally, various examples demonstrate the performance and versatility of our approach.

3. Mechanics of Thin Shells

In the remainder of this paper, vectors and matrices will always be denoted by bold letters, while scalar values are set in italics.

To derive the behaviour of a thin shell undergoing deformation, we start from the potential elastic energy. The thin



(a) (b)

Figure 1: *(a) Sampling the surface with fibers. The fibers passing through the point* $\mathbf{p}$ *measure arc length and curvature along their path. (b) Area approximation using two arc length samples from the fibers* f_1 *and* f_2*. Each fiber estimates the area of the sample according to its own length only. The area represented by* $\mathbf{p}$ *is the average of the fiber areas.*

shell we want to animate is given by a surface Ω, with first fundamental tensor $\mathbf{R}$ and shape operator $\mathbf{S}$. We will follow [TPBF87], and define the potential energy as

$$E_{\text{pot}} = E_s + E_b = \int_\Omega U_s + U_b \, \mathrm{d}\Omega, \tag{1}$$

where E_s and E_b are bending and stretching energies and U_b and U_s are the respective energy densities. For any point $\mathbf{p} \in \Omega$, the energy densities are defined in terms of first fundamental tensor $\mathbf{R}$ and shape operator $\mathbf{S}$:

$$U_s = \frac{K_s}{2} \|\mathbf{R_p} - \mathbf{R}^0_\mathbf{p}\|^2_p, \tag{2}$$

$$U_b = \frac{K_b}{2} \|\mathbf{S_p} - \mathbf{S}^0_\mathbf{p}\|^2_p. \tag{3}$$

Here, the superscript 0 denotes the undeformed (rest) value. To measure difference in shape, we use a pseudo-norm $\|\cdot\|_p$. The parameters K_b and K_s are stiffness constants for bending and stretching deformations respectively. The first fundamental tensor $\mathbf{R}$ measures differential area, while $\mathbf{S}$ measures curvature.

In (1), we have split the potential energy into two distinct components. This effectively denies the existence of mixed bending/stretching modes, which is a common and reasonable assumption [TPBF87, GHDS03].

The core idea of our method is to sample the surface at distinct points and directions. First, we discretize (2) and (3) on a set $P \subset X$ of simulation nodes on the surface. At each sample location, a set of parametric curves is fitted to the surface. These *fibers* measure curvature and arc length in their respective directions. Figure 1 (a) illustrates the sampling.

3.1. Stretching Energy

The stretching term of the potential energy, E_s, measures changes in surface area. In the discrete model, the point $\mathbf{p}$ represents a small surface patch.

We use the fibers through $\mathbf{p}$ to approximate a surface area for $\mathbf{p}$. For each fiber, we assume that the surface patch around $\mathbf{p}$ is circular. A fiber f_k thus measures an area of $A_k = \pi/4 \cdot l(f_k)^2$, where $l(f_k)$ is the arc length of the fiber. The area $A_\mathbf{p}$ of the point $\mathbf{p}$ is defined as the average of these area measurements.

$$A_\mathbf{p} \approx \frac{1}{n_\mathbf{p}} \sum_k A_k = \frac{\pi}{4 n_\mathbf{p}} \sum_k l(f_k)^2, \tag{4}$$

where all fibres f_k pass through $\mathbf{p}$, and $n_\mathbf{p}$ is the number of fibers intersecting in $\mathbf{p}$. See Figure 1 (b) for an illustration.

Note that area approximations for all sample points do not necessarily add up to the total area of the surface Ω. This is not a problem during the simulation. Discretisation effects are discussed in detail in section 8.

We define the stretching energy such that it becomes nonzero when either the total area $A_\mathbf{p}$ or the individual fibre areas A_k change. This amounts to preserving the area represented by each point, as well as the shape of the area element.

$$\tilde{E}_s(\mathbf{p}) = \frac{K_s}{2} \left[\left(A_\mathbf{p} - A^0_\mathbf{p} \right)^2 + \frac{1}{n_\mathbf{p}} \sum_k \left(A_k - A^0_k \right)^2 \right]. \tag{5}$$

3.2. Bending Energy

The shape operator measures curvature: $Tr(\mathbf{S}) = 2H$, where H denotes the mean curvature and $Tr(\cdot)$ is the matrix trace. Noting that the trace is a pseudo-norm as well as a linear operator, we can write

$$\tilde{U}_b = \frac{K_b}{2} \left(H - H^0 \right)^2. \tag{6}$$

For a sample point $\mathbf{p}$, we approximate $H_\mathbf{p}$ using the directional curvature samples given by the fibers through $\mathbf{p}$. We can express $H_\mathbf{p}$ in terms of the normal curvature $\kappa(f_k)$ for each fiber f_k:

$$H_\mathbf{p} \approx \frac{1}{n_\mathbf{p}} \sum_k \kappa(f_k). \tag{7}$$

To obtain the bending energy, we integrate the energy density over the surface element represented by a sample point. Assuming that the energy density is constant over the surface element, the bending energy becomes

$$\tilde{E}_b(\mathbf{p}) = \tilde{U}_{\mathbf{p}b} A^0_\mathbf{p} \tag{8}$$

3.3. Curvature and Arc Length Measurement

In order to evaluate (4) and (7), we need to compute $l(f)$ and $\kappa(f)$. In our implementation, fibers are natural cubic splines through three points on the surface.

Computing the arc length of a cubic spline involves solving an elliptic integral [BJ99]. Therefore, we use numeric integration to approximate the arc length, yielding a function $\tilde{l}(f)$. The approximation is sufficiently accurate even with a very small number of sample points. It is described in detail in Appendix A.

As a measure for curvature, we use the angle θ between the tangents at the start and end point of the fiber. In order to avoid flipping problems, we use a directed angle, which is oriented according to the surface normal $\mathbf{N}$ in the center point of the fiber. Please refer to Appendix A for details.

4. Dynamic Behaviour

We now have all prerequisites to analyze the dynamic behavior of a model sampled with fibers. In order to animate the thin shell, we need to compute forces for the simulation nodes. For one simulation node, the force is given by the negative gradient of the potential energy:

$$\mathbf{F}_i = -\nabla_{\mathbf{p}_i} (\tilde{E}_s + \tilde{E}_b) \tag{9}$$

In our discrete setting, the force on simulation node i can be written as a sum over the contributions of all simulation nodes.

$$\mathbf{F}_i = \sum_j \mathbf{F}_{ji} = -\sum_j \left(\nabla_{\mathbf{p}_i} \tilde{E}_s(\mathbf{p}_j) + \nabla_{\mathbf{p}_i} \tilde{E}_b(\mathbf{p}_j) \right) \tag{10}$$

Substituting (5) and (8) into (10) and evaluating the gradients, we obtain

$$\begin{aligned} \mathbf{F}_{ji} \;=\; & -\frac{K_s}{n_{\mathbf{p}}} \sum_k \left(A_k - A_k^0 \right) \nabla_{\mathbf{p}_i} A_k \\ & -K_s \left(A_{\mathbf{p}_j} - A_{\mathbf{p}_j}^0 \right) \nabla_{\mathbf{p}_i} A_j \\ & -A_j^0 K_b \left(H_j - H_j^0 \right) \nabla_{\mathbf{p}_i} H_j \end{aligned} \tag{11}$$

Noting that both $\nabla_{\mathbf{p}_i} A_{\mathbf{p}_j}$ and $\nabla_{\mathbf{p}_i} H_j$ are sums over the same fibers, we can split the force $\mathbf{F}_{ji}$ into components induced by individual fibers f_k.

$$\mathbf{F}_{ji} \;=\; \sum_k \mathbf{F}_{ki} \tag{12}$$

$$\begin{aligned} \mathbf{F}_{ki} \;=\; & -\frac{K_s}{n_{\mathbf{p}}} \left(A_{\mathbf{p}_j} + A_k - A_{\mathbf{p}_j}^0 - A_k^0 \right) \nabla_{\mathbf{p}_i} A_k \\ & -A_j^0 \frac{K_b}{n_{\mathbf{p}}} \left(H_j - H_j^0 \right) \nabla_{\mathbf{p}_i} \kappa(f_k) \end{aligned} \tag{13}$$

$\mathbf{F}_{ki}$ is the force that the fiber k, running through the point $\mathbf{p}_j$ exerts on the point $\mathbf{p}_i$. The gradient of one fiber's area estimate A_k is given by

$$\nabla_{\mathbf{p}_i} A_k = \frac{\pi}{2} \tilde{l}(f_k) \nabla_{\mathbf{p}_i} \tilde{l}(f_k). \tag{14}$$

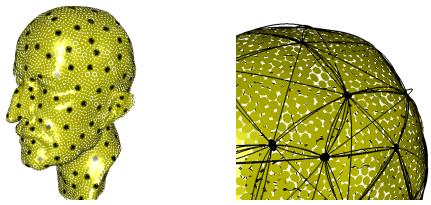
Expressions for the gradients of $\tilde{l}(f)$ and $\kappa(f)$ are stated in Appendix A. Taking all surface elements into consideration, the resulting forces add up to zero.

We compute forces in two passes. In a first pass, we iterate over all fibers to compute $\tilde{l}(f)$ and $\kappa(f)$ and their respective gradients. In a second pass over all simulation nodes, we can evaluate (13) and sum the force contributions to obtain $\mathbf{F}_i$.

The governing equation for our system is

$$\mathbf{F} = -\nabla_{\mathbf{x}} E_{\text{pot}} = \mathbf{M}\ddot{\mathbf{x}} + \eta\dot{\mathbf{x}} + \mathbf{F}_{\text{ext}} \tag{15}$$

where $\mathbf{M}$ is the mass matrix, η is a damping coefficient, and



(a) (b)

Figure 2: *(a) The sampling of the Max Planck model with simulation nodes. (b) Fiber network on the surface.*

$\mathbf{F}_{\text{ext}}$ denotes the external forces acting on the system. The system state $\mathbf{x}$ contains the position of all simulation nodes.

Upon computing $\mathbf{F}_i$ for all simulation nodes, we use the velocity Verlet integration scheme [Ver67] to solve (15) and animate the model.

5. Fiber Creation

For a surface Ω which we animate using our thin shell approach, we require a set of sample points or simulation nodes $P \subset \Omega$. The sampling does not need to be regular, but it should be adequate in a sense that all features of the surface need to be represented with sample points [PKKG03].

The simulation nodes serve as end points for fibers, and carry the mass of the model. Usually, the model is rendered using a different surface which is deformed along with the model (see Section 6). Figure 2 shows the sampling of a model with simulation nodes and fibers.

The fibers sample a three dimensional space: two dimensions for the position on the surface, plus one dimension representing the direction of the fibers. We have to make sure that all dimensions are adequately sampled. Therefore we enforce a roughly isotropic sampling in the directional domain. Section 7.3 deals with issues related to representing anisotropy or inhomogenities in the simulated material.

We use a simple heuristic to create an isotropic sampling. For a node $\mathbf{p}$, let $P_{\mathbf{p}} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \subset P$ be the set of nearest neighbors to $\mathbf{p}$. For each of the $\mathbf{p}_i$, we find the most opposite point with respect to $\mathbf{p}$, $\mathbf{p}_j$, i.e.

$$\mathbf{p}_j = \arg\min_{\mathbf{q} \in P_{\mathbf{p}}} \langle \mathbf{q} - \mathbf{p}, \mathbf{p} - \mathbf{p}_i \rangle \tag{16}$$

We then create a fiber connecting $\mathbf{p}_i, \mathbf{p}, \mathbf{p}_j$. Duplicate fibers are discarded. The equations for the parametric curve that is created for the three input points are given in Appendix A. Figure 3 illustrates the process.

On average, this heuristic approach creates an isotropic sampling of the surface. There are of course pathological cases in which this heuristic leads to degenerate sampling. However, since we can control which points on the surface become simulation nodes, these cases can be avoided. We use an adapted version of the clustering algorithm described in [PGK02] to select a set of simulation nodes.

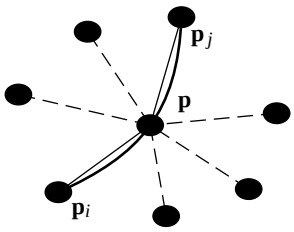


Figure 3: *Fiber creation. In a neighborhood of nodes $P_\mathbf{p}$ around a central point $\mathbf{p}$, one fiber is created for each $\mathbf{p}_i \in P_\mathbf{p}$, connecting $\mathbf{p}_i$, $\mathbf{p}$, and $\mathbf{p}_j$, which is the most opposite node for $\mathbf{p}_i$.*

6. High-Resolution Surface

For interactive or real-time applications, the computational effort of simulating the entire model is too high. In order to be able to simulate visually appealing models with highly detailed surfaces in reasonable time, we sample the surface coarsely and use only these samples as simulation nodes.

We deform the original surface of the model by interpolating the deformation field defined only in the simulation nodes. For each *passive point* on the surface that needs to be deformed, we compute local coordinates with respect to its neighborhood of simulation nodes. When the object deforms and the simulation nodes move, an updated position of the passive point can be computed using these local coordinates.

This method is similar to the skinning approach proposed by Singh and Kokkevis in [SK00]. While they compute barycentric coordinates and a normal displacement on a triangle mesh, we use coordinate systems based on two simulation nodes.

Let $\mathbf{c}$ be the position of a passive point and $P_c = \{\mathbf{p}_i, i = 0 \ldots n\}$ for some n a set of neighboring simulation nodes in their undeformed state, ordered such that $\|\mathbf{p}_i - \mathbf{c}\| < \|\mathbf{p}_{i+1} - \mathbf{c}\|$. We first compute a least squares plane through the $\mathbf{p}_i$. The normalized plane normal will be denoted $\mathbf{N}$. We then transform $\mathbf{c} - \mathbf{p}_0$ into n coordinate systems $\{\mathbf{e}_i^1, \mathbf{e}_i^2, \mathbf{e}_i^3\}$, yielding coordinates $\mathbf{C}_i$. The coordinates systems are computed from the normal of the least-squares plane and the position of one of the neighbors relative to the nearest neighbor:

$$\mathbf{e}_i^1 = \mathbf{p}_i - \mathbf{p}_0 \qquad \mathbf{e}_i^2 = \frac{\mathbf{N} \times \mathbf{e}_i^1}{\|\mathbf{e}_i^1\|} \qquad \mathbf{e}_i^3 = \frac{\mathbf{e}_i^2 \times \mathbf{e}_i^1}{\|\mathbf{e}_i^1\|} \tag{17}$$

Each of these coordinate systems measures surface stretch in the direction $\mathbf{e}_i^1$, and account for rotations around $\mathbf{e}_i^1$. Due to the normalization of $\mathbf{e}_i^2$ and $\mathbf{e}_i^3$, the coordinate system i does not capture surface stretch in these directions.

Local coordinates are computed as

$$C_i^k = \frac{\left\langle \mathbf{c} - \mathbf{p}_0, \mathbf{e}_i^k \right\rangle}{\|\mathbf{e}_i^k\|^2} \tag{18}$$

and are stored with the passive point. When the surface deforms, we transform the local coordinates back into world coordinates, yielding n positions $\mathbf{c}_i'$.

$$\mathbf{c}_i' = \mathbf{p}_0' + C_i^1 \cdot \mathbf{e}_i^{1\prime} + C_i^2 \cdot \mathbf{e}_i^{2\prime} + C_i^3 \cdot \mathbf{e}_i^{3\prime} \tag{19}$$

Here, the coordinate systems $\mathbf{e}_i^{k\prime}$ are computed from the deformed simulation node positions $\mathbf{p}_i'$. The deformed position $\mathbf{c}'$ is a computed as a weighted sum of the positions $\mathbf{c}_i'$.

$$\mathbf{c}' = \frac{\sum_i w_i \mathbf{c}_i'}{\sum_i w_i} \tag{20}$$

and

$$w_i = w\left(\frac{\|\mathbf{p}_i - \mathbf{c}\|}{\|\mathbf{p}_n - \mathbf{c}\|}\right) \tag{21}$$

Using some weight function $w(\cdot)$ satisfying $w(0) = \infty$ and $w(1) = 0$. Thus, the surface deformation is smooth provided that the weight function is smooth.

Note that any surface representation can be used as a high-resolution surface using this deformation method.

7. Plasticity and Fracturing

So far, we have only treated purely elastic objects. However, most objects exhibit plasticity, or fracture under high stress. This section describes how plasticity and fracturing can be integrated into our framework.

7.1. Plasticity

Materials that exhibit plastic behaviour remember part of their deformation. Elastic forces will not restore the original shape, but a new shape created by plastic deformation. In our model, the rest shape of an object is stored in the rest shape of the fibers used to sample it. It is thus sufficient to consider a single fiber.

The elastic force for a fibre is computed using its rest arc length l^0 and rest tangential angle θ^0, as well as the current length l and current tangential angle θ. To incorporate plasticity, the rest length and rest angle used to compute the elastic force is expressed in terms of the original value and a plastic deformation

$$\begin{aligned} l^0 &= l_{\text{orig}} + l_{\text{plastic}} \\ \theta^0 &= \theta_{\text{orig}} + \theta_{\text{plastic}} \end{aligned} \tag{22}$$

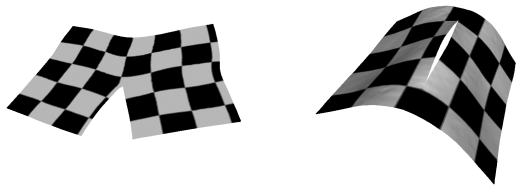
The plastic deformation represented by l_{plastic} and θ_{plastic} changes whenever the deformation of the fiber becomes too high. The rest arc length of a fiber changes if

$$\beta_l < |l - l^0|, \tag{23}$$

where β_l is the plastic yield constant for stretching deformations. Similarly, the curvature θ_{plastic} is updated if

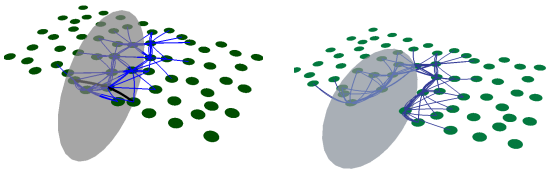
$$\beta_\theta < |\theta - \theta^0|, \tag{24}$$

where β_θ is the plastic yield threshold for bending deformations. We update the rest state similar to [OBH02], however



(a) (b)

Figure 4: *Different fracture modes. A crack introduced by (a) tensile stress; (b) bending stress.*



(a) (b)

Figure 5: *Fracturing. (a) When stress along a fiber exceeds a fracturing threshold (black), a new crack plane orthogonal to the fiber tangent is created. (b) Fiber sampling after neighborhood recomputation. Fibers intersecting the crack plane have been deleted, the edges have been resampled.*

accounting for plastic creep by adding time dependence to the update rule. In each timestep in which (23) or (24) hold, l_{plastic} and θ_{plastic} are changed according to (25) and (26) respectively.

$$\Delta l_{\text{plastic}} = \eta_l (l - l_0) \Delta t \quad (25)$$

$$\Delta \theta_{\text{plastic}} = \eta_\theta (\theta - \theta_0) \Delta t \quad (26)$$

The plastic creep parameters $\eta_{l,\theta}$ determine how fast the material can adapt to the new state. In addition to this basic form of plasticity, a variety of other nonlinear effects can be modeled, an example is given in Section 7.4.

7.2. Fracturing

When a material fractures or tears, parts of the material that are separated by a crack do not influence each other, even though they might be close. This coupling between simulation nodes is modeled by fibers in our framework. Thus, in order to model a crack, we have to cut the fibers that intersect the crack surface.

We initiate a crack if the stress on some fiber in the model becomes too high, specifically if one of

$$\begin{aligned} \gamma_l &< l/l^0 \\ \gamma_\theta &< |\theta - \theta^0| \end{aligned} \quad (27)$$

holds for a fiber f. γ_l and γ_θ are fracture thresholds for tensile stretching and bending fracture respectively. We do not model compressive stress. See Figure 4 for an illustration.

Fracturing is a sudden and violent process, introducing discontinuities into the model. As such, it poses extreme difficulties for any simulation based on discrete timesteps. In order to alleviate these problems, we only allow one crack to form in each timestep. Furthermore, if in one timestep a crack has formed, fracturing is disabled for a number of simulation steps, until the material had enough time to relax and dissipate the freed energy.

If the material is under high stress, the condition (27) is usually fulfilled for several fibers. Of these, we choose the fiber where fracture is most urgent, i.e. the fiber wich maximizes

$$\max\left(\frac{l/l^0}{\gamma_l}, \frac{|\theta - \theta^0|}{\gamma_\theta}\right) \quad (28)$$

Once a fiber is chosen, a small disc, the crack plane, is placed at the center node $\mathbf{p}_c$ of the fiber. Its normal is the fiber's tangent in $\mathbf{p}_c$, and its radius is equal to the average node distance in the neighborhood. The neighborhoods of nearby simulation nodes are recomputed accounting for crack planes, i.e. only nodes that are not separated by a crack plane can be in the same neighborhood. We then reinitialize fibers as described in Section 5, using the original positions of the simulation nodes, but the newly computed neighborhoods. This effectively cuts any fibres that would intersect with a crack plane. The recomputation of neighborhoods makes sure that the material remains adequately sampled with fibers, also along the crack surfaces. Figure 5 illustrates the procedure. For surface points in the vincinity of a new crack, the neighborhoods and coordinates for the high resolution surface have to be recomputed as well.

We can model microcracks and material weaknesses introduced by fracture effects by only creating a crack plane if more than a certain ratio of the fibres passing through a node have been broken. If the stress on a fiber exceeds a fracturing threshold, we consider this fiber broken. Its contribution to the forces acting on connected nodes will from now on be zero.

Controlled fracture can be easily implemented by assigning very low fracture thresholds in regions where cracks are desired and high thresholds where the material should not fracture.

7.3. Anisotropy and Inhomogenity

Many materials exhibit anisotropic behaviour, for example stiffness varying depending on stress direction. We found that the best way to integrate anisotropy is to use a roughly regular and isotropic sampling throughout the model, and vary the stiffness constants of the individual fibers according to position or direction, or time.

7.4. Nonlinear Effects

Many materials change their material properties depending on the current strain state or the amount of plastic deforma-

tion they undergo. It is easy to integrate such effects into our framework. Since all material properties are stored with the individual fibers, we can vary these parameters depending on the current state of the fiber (the local strain state of the material), or the past deformations of the fiber.

As an example, the following update rule for the fracture thresholds implements a behaviour often found in hard plastics and metals. The material weakens under repeated plastic bending deformation.

$$\Delta\gamma_\theta = -\alpha\gamma_\theta|\Delta\theta_{\text{plastic}}| \tag{29}$$

Here, the parameter α determines how much the material is weakened by plastic deformation. $\Delta\theta_{\text{plastic}}$ is the change of plastic deformation as defined in (26), which is evaluated in each time step.

8. Discussion

The discretisation approach described in sections 3 and 4 will lead to different material behaviour depending on the sampling chosen. Since the forces are normalized, increasing the number of fibers per point does not increase material stiffness, however, anisotropic sampling results in anisotropic material behaviour. This occurs in regions of highly irregular sampling, where the heuristic described in Section 5 breaks down.

Under very large stretching deformations, the initial sampling degrades. Currently, no resampling of the simulation nodes is done. Hence, simulation quality suffers after applying a large (plastic) deformation.

In our current implementation, we solve the differential equation governing the thin shell behaviour using explicit time integration. While being fast, this integration method is not stable under arbitrary conditions. For very stiff materials, oscillations and instabilities occur, especially if discontinuous processes such as fracture are considered. Simulation timesteps have to be extremely small, and simulation times can become prohibitive. In order to simulate a wider range of materials, an implicit integration scheme has to be implemented.

As input, our methods expects a set of sample points that adequately represent the surface. While we implemented our framework using point-sampled surfaces, the simulation method will work for other surface representation as well. In order to use a different input representation, the required sample points on the surface are generated before the fiber network is initialized. A simple sampling algorithm can be used to provide the samples [LR98]. The surface deformation method described in Section 6 is applicable to other surface representations as well [SK00].

Although the fiber-based simulation avoids creating a consistent triangulation, connectivity is explicitly encoded in the fibers. Conceptually, a truly meshless approach, similar to [MKN*04], would be more elegant.

9. Results

We have tested our simulation method on a variety of models. Simulation times were measured in a 3 GHz PC. Table 1 summarizes the measurements. Shown are the number of simulation nodes, the number of surface points, as well as average simulation times per frame. As can be seen from the data, the animation of the high-resolution surface is relatively expensive. Since it is only performed once per frame, it is a good alternative to a full simulation if the simulation timestep is small compared to the frame time. For extremely slow motion animations, such as the balloon sequence, surface reconstruction is clearly the limiting factor.

			average time/frame [ms]			
Model	#Nodes	#Points	forces	fract.	surf.	total
Max	931	9835	7	—	64	71
Balloon	1170	24578	24	23	166	212
Mask	1987	40880	222	—	325	547
Pinup	11860	240000	1533	6066	1459	9058

Table 1: *Simulation times for several models. Shown are (from left to right): number of simulation nodes; number of surface points; time for force computation and integration; time for fracture testing and neighborhood recomputation; time for high-resolution surface animation; total animation time per frame.*

Figure 6 presents frames from an animation showing an elastic mask dropping on the floor. We also dropped a bronze bust to demonstrate plasticity. Figure 7 shows some frames from the resulting animation. Plasticity has no significant impact on simulation times.

In Figure 8, a poster is torn apart. We placed a weak point in the middle of the top edge to initiate the tearing process. The material has a very high stretching stiffness, leading to oscillations along the tear. The discontinuities introduced by the tearing process put extreme strain on the explicit integration.

Figure 9 shows frames from an animation in which a balloon tears as it is inflated too much. As can be seen from Table 1, fracturing is quite expensive.

Please also refer to the accompanying videos for the complete animations.

10. Conclusion

We have presented a novel framework for the simulation of thin shells. Using a sampling approach to capture the geometric surface properties, we are able to compute the forces derived from curvature and arc length. Since the simulation nodes can be chosen at arbitrary locations on the surface, the method is mostly independent of the initial sampling of the surface, as well as the surface representation. The proposed simulation method can handle a wide range of mate-

rials, from pure elastic or plastic objects to materials showing ductile or brittle fracture. We also devised a method to smoothly interpolate the displacement field defined only at the simulation nodes in order to transfer the computed deformation to a high resolution surface.

Future research will focus on implementing an implicit integration scheme for the simulation of extremely stiff materials. For large deformations, a resampling strategy is needed that keeps the surface adequately sampled.

References

[Bat95] BATHE K. J.: *Finite Element Procedures*. Prentice-Hall, 1995. 1

[BFA02] BRIDSON R., FEDKIW R., ANDERSON J.: Robust Treatment of Collisions, Contact and Friction for Cloth Animation. In *Proceedings of SIGGRAPH* (2002). 1, 2

[BJ99] BANCSIK Z., JUHÁSZ I.: On the Arc Length of Parametric Cubic Curves. *Journal for Geometry and Graphics 3*, 1 (1999), 1–15. 3

[BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of Clothing with Folds and Wrinkles. In *Proceedings of the Symposium on Computer Animation* (2003). 1, 2

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of SIGGRAPH* (1998), pp. 43–54. 2

[CG91] CELNIKER G., GOSSARD D.: Deformable Curve and Surface Finite-Elements for Free-Form Shape Design. In *Proceedings of SIGGRAPH* (1991), pp. 257–266. 2

[COS00] CIRAK F., ORTIZ M., SCHRÖDER P.: Subdivision Surfaces: A New Paradigm for Thin-Shell Finite-Element Analysis. *Int. J. for Numerical Methods in Engineering 47* (2000), 2039–2072. 2

[GBO04] GOKTEKIN T., BARGTEIL A. W., O'BRIEN J. F.: A Method for Animating Viscoelastic Fluids. In *Proceedings of SIGGRAPH* (2004), pp. 463–467. 1

[GHDS03] GRINSPUN E., HIRANI A. N., DESBRUN M., SCHRÖDER P.: Discrete shells. In *Proceedings of the Symposium on Computer Animation* (2003), pp. 62–67. 2, 3

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: CHARMS: A Simple Framework for Adaptive Simulation. In *Proceedings of SIGGRAPH* (2002). 2

[GTS02] GREEN S., TURKIYYAH G., STORTI D.: Subdivision-Based Multilevel Methods for the Large Scale Simulation of Thin Shells. In *Proceedings of Solid Modeling and Applications* (2002), pp. 265–272. 2

[GZ68] GREEN A. E., ZERNA W.: *Theoretical Elasticity*. Oxford University Press, 1968. 1

[KGC*96] KOCH M., GROSS M., CARLS F., VON DÜREN D., FRANKHAUSER G., PARISH Y.: Simulating Facial Surgery Using Finite Element Models. In *Proceedings of SIGGRAPH* (1996), pp. 421–428. 1, 2

[LR98] LISCHINSKY D., RAPOPORT A.: Image-based rendering for non-diffuse synthetic scenes. In *Proceedings of the Eurographics Workshop on Rendering Techniques* (1998), pp. 301–314. 7

[MBF04] MOLINO N., BAO Z., FEDKIW R.: A Virtual Node Algorithm for Changing Mesh Topology During Simulation. In *Proceedings of SIGGRAPH* (2004). 2

[MDSB02] MEYER M., DESBRUN M., SCHRÖDER P., BARR A.: Discrete Differential Geometry Operators for Triangulated 2-Manifolds. VisMath, 2002. 2

[MG04] MÜLLER M., GROSS M.: Interactive Virtual Materials. In *Proceedings of Graphics Interface* (2004), pp. 239–246. 1

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., ALEXA M., GROSS M.: Point Based Animation of Elastic, Plastic and Melting Objects. In *Proceedings of the Symposium on Computer Animation* (2004), pp. 141–151. 2, 7

[OBH02] O'BRIEN J. F., BARGTEIL A. W., HODGINS J. K.: Graphical modeling and animation of ductile fracture. In *Proceedings of SIGGRAPH* (2002), pp. 291–294. 5

[OH99] O'BRIEN J. F., HODGINS J. K.: Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH* (1999), pp. 137–146. 1

[PGK02] PAULY M., GROSS M. H., KOBBELT L.: Efficient Simplification of Point-Sampled Surfaces. In *Proceedings of Visualization* (2002), pp. 163–170. 4

[PKKG03] PAULY M., KEISER R., KOBBELT L., GROSS M.: Shape Modeling with Point-sampled Geometry. In *Proceedings of SIGGRAPH* (2003), pp. 641–650. 2, 4

[SK00] SINGH K., KOKKEVIS E.: Skinning Characters using Surface-Oriented Free-Form Deformations. In *Proceedings of Graphics Interface* (2000), pp. 35–42. 5, 7

[ST92] SZELISKI R., TONNESEN D.: Surface modeling with oriented particle systems. In *Proceedings of SIGGRAPH* (1992), pp. 185–194. 2

[THMG04] TESCHNER M., HEIDELBERGER B., MÜLLER M., GROSS M.: A Versatile and Robust Model for Geometrically Complex Deformable Solids. In *Proceedings of Computer Graphics International* (2004), pp. 312–319. 2

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically Deformable Models. In *Proceedings of SIGGRAPH '87* (1987), pp. 205–214. 1, 2, 3

[Ver67] VERLET L.: Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Physical Review 159*, 1 (1967), 98–103. 4

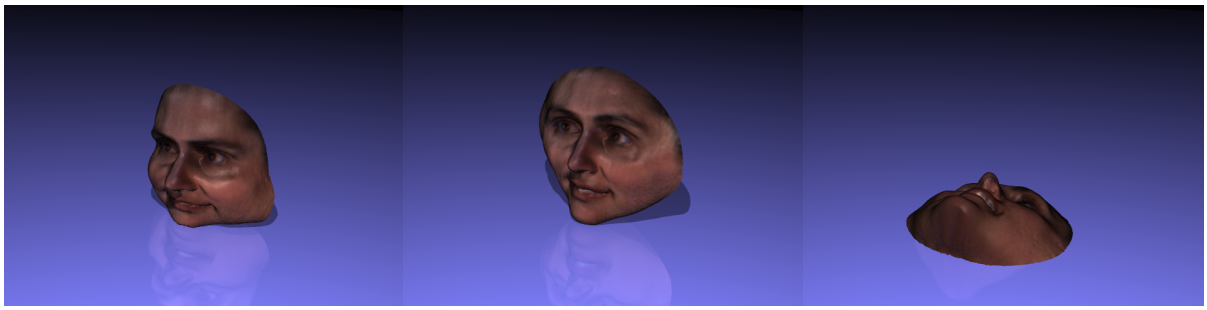


Figure 6: *An elastic mask is dropped. The shell deforms and bounces, before coming to a rest state on the surface.*

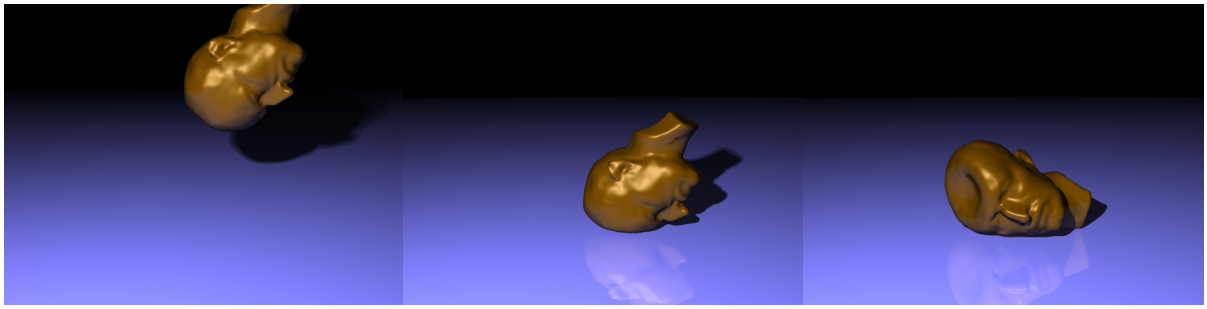


Figure 7: *A heavy bronze bust is dropped on the floor. Note that the expected behaviour for a solid object would be to form a flat surface where plastically deformed by the impact. However, due to the lack of volume preservation forces, a shell caves in.*

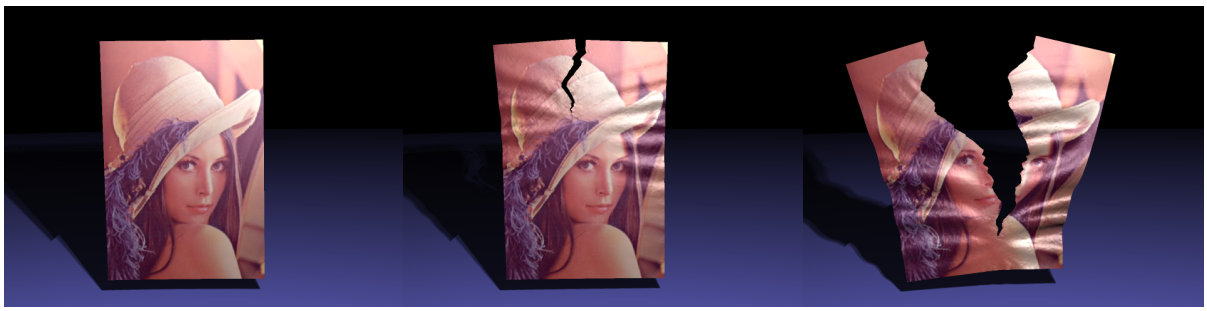


Figure 8: *A large poster is torn apart by external forces. We start the crack by adding a weak point to the top edge of the sheet. Since the material is not brittle and the forces are moderate, the crack propagates slowly.*

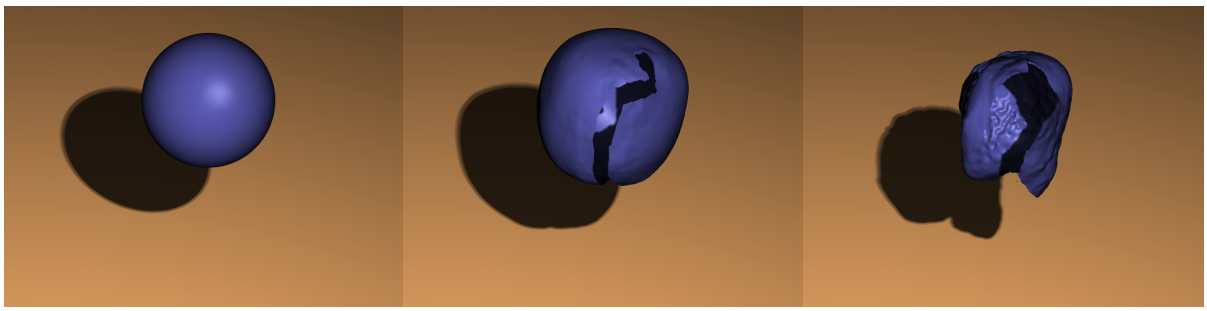


Figure 9: *A balloon tears when inflated too much. Once the tear is opened, the shell can relax and the tearing process stops.*

Appendix A:

Spline Equations

A fiber in our implementation is a natural cubic spline through three simulation nodes on the object surface. Its parametric representation is given as a piecewise cubic polynomial:

$$f(t) = \begin{cases} f_1(2t) & 0 <= t < 0.5 \\ f_2(2(t-0.5)) & 0.5 <= t < 1 \end{cases} \quad (30)$$

for x, y, and z respectively. The polynomials f_1 and f_2 are defined as

$$f_i(t) = a_i + b_i t + c_i t^2 + d_i t^3, \quad (31)$$

their coefficients depend linearly on the input points $\mathbf{p}_1$, $\mathbf{p}$ and $\mathbf{p}_2$:

$$\begin{array}{rcl} a_1 &=& \mathbf{p}_1 \\ a_2 &=& \mathbf{p} \\ b_1 &=& -\frac{5}{4}\mathbf{p}_1 + \frac{3}{2}\mathbf{p} - \frac{1}{4}\mathbf{p}_2 \\ b_2 &=& -\frac{1}{2}\mathbf{p}1 + \frac{1}{2}\mathbf{p}_2 \\ c_1 &=& 0 \\ c_2 &=& \frac{3}{4}\mathbf{p}_1 - \frac{3}{2}\mathbf{p} + \frac{3}{4}\mathbf{p}_2 \\ d_1 &=& \frac{1}{4}\mathbf{p}_1 - \frac{1}{2}\mathbf{p} + \frac{1}{4}\mathbf{p}_2 \\ d_2 &=& -\frac{1}{4}\mathbf{p}_1 + \frac{1}{2}\mathbf{p} - \frac{1}{4}\mathbf{p}_2 \end{array} \quad (32)$$

Arc Length Approximation $\tilde{l}(f)$

The arc length of a parametric curve can be numerically approximated by

$$\tilde{l}(f) = \frac{1}{n}\sum_{i=0}^{n-1} \| f'(\frac{i}{n}) \| \quad (33)$$

Taking the first derivative of (31) and coefficients defined in (32), (33) can be computed easily. Since the splines in our framework are generally well-behaved, the approximation is sufficiently accurate even for a small number of sample points.

Gradient of $\tilde{l}(f)$

The gradient of (33) is essentially a sum of gradients.

$$\nabla\tilde{l}(f) = \frac{1}{n}\sum_{i=0}^{n-1} \nabla\| f_1'(\frac{i}{n})\| + \frac{1}{n}\sum_{i=0}^{n-1} \nabla\| f_2'(\frac{i}{n})\| \quad (34)$$

Thus, the gradients of the arc length with respect to the center point $\mathbf{p}$ and the endpoints $\mathbf{p}_1$ and $\mathbf{p}_2$ can be written as

$$\begin{aligned} \nabla_{\mathbf{p}_1}\tilde{l}(f) &= \sum_{i=0}^{n-1} \frac{1}{n\| f_1'(i/n)\|}(-\frac{5}{4} + \frac{3}{4}t^2) f_1'(i/n) \quad (35) \\ &\quad + \sum_{i=0}^{n-1} \frac{1}{n\| f_2'(i/n)\|}(-\frac{1}{2} + \frac{3}{2}t - \frac{3}{4}t^2) f_2'(i/n) \\ \nabla_{\mathbf{p}}\tilde{l}(f) &= \sum_{i=0}^{n-1} \frac{1}{n\| f_1'(i/n)\|}(\frac{3}{2} - \frac{3}{2}t^2) f_1'(i/n) \quad (36) \\ &\quad + \sum_{i=0}^{n-1} \frac{1}{n\| f_2'(i/n)\|}(-3t + \frac{3}{2}t^2) f_2'(i/n) \\ \nabla_{\mathbf{p}_2}\tilde{l}(f) &= \sum_{i=0}^{n-1} \frac{1}{n\| f_1'(i/n)\|}(-\frac{1}{4} + \frac{3}{4}t^2) f_1'(i/n) \quad (37) \\ &\quad + \sum_{i=0}^{n-1} \frac{1}{n\| f_2'(i/n)\|}(\frac{1}{2} + \frac{3}{2}t - \frac{3}{4}t^2) f_2'(i/n) \end{aligned}$$

Tangential Angle θ_f

As a measure for the curvature of a fiber f, we use its tangential angle θ_f, the angle between the fiber's tangents at $t = 0$, $\mathbf{t}_0$ and $t = 1$, $\mathbf{t}_1$. This angle is oriented according to the surface normal in $\mathbf{p}$. Therefore, we multiply the angle with the sign of $\langle \mathbf{N}, f''(0.5) \rangle$.

$$\kappa(f) = \theta_f = \text{sign}(\langle \mathbf{N}, f''(0.5) \rangle) \arccos \frac{\langle \mathbf{t}_0, \mathbf{t}_1 \rangle}{\|\mathbf{t}_0\| \|\mathbf{t}_1\|} \quad (38)$$

Gradient of $\kappa(f)$

Since the derivative of signum is zero, the gradient of (33) with respect to a point $\mathbf{p}_i$ is

$$\nabla_{\mathbf{p}_i}\kappa(f) = \text{sign}(\langle \mathbf{N}, f''(0.5) \rangle) \nabla_{\mathbf{p}_i} (\arccos \frac{\langle \mathbf{t}_0, \mathbf{t}_1 \rangle}{\|\mathbf{t}_0\| \|\mathbf{t}_1\|}) \quad (39)$$

$\mathbf{t}_0 = b_1$ and $\mathbf{t}_1 = b_2 + 2c_2 + 3d_2$. Thus, for center point $\mathbf{p}$ and endpoints $\mathbf{p}_1$ and $\mathbf{p}_2$,

$$\begin{aligned} \nabla_{\mathbf{p}_1}\kappa(f) &= c(\tfrac{-\frac{9}{2}d_1 - \mathbf{t}_c}{\|\mathbf{t}_0\|\|\mathbf{t}_1\|} + \tfrac{5}{4}\tfrac{\langle \mathbf{t}_0, \mathbf{t}_1 \rangle}{\|\mathbf{t}_0\|^3 \|\mathbf{t}_1\|}\mathbf{t}_0 - \tfrac{1}{4}\tfrac{\langle \mathbf{t}_0, \mathbf{t}_1 \rangle}{\|\mathbf{t}_0\|\|\mathbf{t}_1\|^3}\mathbf{t}_1) \\ \nabla_{\mathbf{p}}\kappa(f) &= c(\tfrac{9d_1}{\|\mathbf{t}_0\|\|\mathbf{t}_1\|} - \tfrac{3}{2}\tfrac{\langle \mathbf{t}_0, \mathbf{t}_1 \rangle}{\|\mathbf{t}_0\|^3 \|\mathbf{t}_1\|}\mathbf{t}_0 + \tfrac{3}{2}\tfrac{\langle \mathbf{t}_0, \mathbf{t}_1 \rangle}{\|\mathbf{t}_0\|\|\mathbf{t}_1\|^3}\mathbf{t}_1) \quad (40) \\ \nabla_{\mathbf{p}_2}\kappa(f) &= c(\tfrac{-\frac{9}{2}d_1 + \mathbf{t}_c}{\|\mathbf{t}_0\|\|\mathbf{t}_1\|} + \tfrac{1}{4}\tfrac{\langle \mathbf{t}_0, \mathbf{t}_1 \rangle}{\|\mathbf{t}_0\|^3 \|\mathbf{t}_1\|}\mathbf{t}_0 - \tfrac{5}{4}\tfrac{\langle \mathbf{t}_0, \mathbf{t}_1 \rangle}{\|\mathbf{t}_0\|\|\mathbf{t}_1\|^3}\mathbf{t}_1) \end{aligned}$$

where $\mathbf{t}_c = b_2$ (i.e. the tangent in the central point $\mathbf{p}$) and

$$c = \frac{\text{sign}(\langle \mathbf{N}, f''(0.5) \rangle)}{\sqrt{1 - \cos^2(\theta)}} \quad (41)$$