A Discrete Model for Inelastic Deformation of Thin Shells

Yotam Gingold Adrian Secord Jefferson Y. Han Eitan Grinspun
Denis Zorin*
Media Research Lab, New York University

August 21, 2004

Abstract

We introduce a method for simulating the inelastic deformation of thin shells: we model plasticity and fracture of curved, deformable objects such as light bulbs, egg-shells and bowls. Our novel approach uses triangle meshes yet evolves fracture lines unrestricted to mesh edges. We present a novel measure of bending strain expressed in terms of surface invariants such as lengths and angles. We also demonstrate simple techniques to improve the robustness of standard timestepping as well as collision-response algorithms.

Introduction

Aluminum cans, dry autumn leaves, and straw hats are everyday examples of *thin shells*: thin, curved, deformable objects. When strained, shells exhibit a broad range of inelastic deformations, *i.e.*, permanent changes in shape such as the plastic deformation of a crushed soda can or the fracture of a crushed dry leaf. While the modeling of plasticity and fracture has long been a goal of the graphics community [TF88], recent successful efforts [OH99] have focused on solids, not thin shells.

Our approach to modeling the inelastic deformation of thin shells follows recent advances in discrete formulations for mechanics [Gre73, MW01, GDHS03] and differential geometry [MDSB03, CSM03]: we represent the shell using an ordinary triangle mesh and formulate strain in terms of quantities which do not depend on a (global or local) coordinate frame, *i.e.*, in terms of *surface invariants* such as edge length, triangle area, and interior- and dihedral-angles.

Contributions. Our main contributions are unified by the concept of *bending strain* and the discretization of bending strain defined over faces of a triangle mesh. This strain measure is simple to compute, captures a continuous range of bending directions, and has a simple expression in terms of face areas, edge lengths, and dihedral angles. Our bending strain is compatible with the usual membrane strain, which makes it possible to treat both in a uniform way.

Our strain discretizations form the foundation for a discrete model for shell plasticity and fracture. To the best of our knowledge, this is the first fracture method for computer animation applications which is formulated for objects represented by triangle meshes while not constraining fractures to run along existing mesh edges.

Furthermore, we present three algorithmic techniques that improve the robustness and quality of our simulations. First, we demonstrate that a simple algorithm *to search for fracture events* improves the robustness of our fracture code. Second, we introduce *vertex budging*, which improves mesh (and animation) quality by subtly reparameterizing the surface during fracture events. Finally, we describe the modifications that make our *collision response* code robust in the presence of fracture events. We demonstrate the benefit of these three improvements in several animations including a punctured sheet, breaking bowl, and shattered lightbulb.

Context

Our work applies techniques based on discrete differential operators recently developed for geometric modeling applications for physically based simulation. Most closely related is the recent work of [GDHS03] which uses a purely geometric approach to deriving a discrete model for elastic thin shells. We build on important recent developments in discrete geometry: a simple formulation for the discrete shape operator [CSM03], used *e.g.*, in [ACSD*03] for anisotropic remeshing and in [HP04] for anisotropic smoothing.

These techniques make it possible to use conventional

*{gingold, ajsecord, jhan, eitan, dzorin}@mrl.nyu.edu

Figure 1: We used our novel measure of discrete strain to model plasticity and fracture of thin shelled objects. Shown are frames from our simulation of the shattering of a lightbulb.

continuum mechanics models for qualitative simulations without incurring prohibitive computational penalties typical, in particular, for finite element simulations of shell models, which remain an active research area in the engineering community (see [COS00] and references therein).

Related work on fracture. Several papers in the computer graphics literature have considered fracture. The fracture of surfaces and solids was first demonstrated in [TF88] who used curvature-controlled splines and later by [NTB*91] who used mass-spring systems. While these works showed examples of thin plates, which are *flat* in their undeformed state, none showed examples of thin shells. Furthermore, in both works the orientation of the fracture line was not determined; rather fracture was effected by removing the connection between vertices. Consequently the resulting fracture lines exhibited aliasing as they followed only existing mesh edges. In contrast, our approach determines the direction of the fracture line and inserts mesh edges as needed. Also related is the procedural approach of [NF99], who used a recursive pattern generator to crack a plane into polygonal shards.

The engineering literature has a large body of work on fracture, however the bulk of this work deals with solids rather than shells. In a concurrent development dealing with shell fracture for engineering simulations, [COPar] use subdivision elements and cohesive edge elements to model fractures occurring along existing mesh edges.

Recently in graphics, Shen and Yang [SY98] used hexahedral volume meshes to simulate deformable objects with fracture. O'Brien and Hodgins [OH99, OBH02] produced appealing finite-element animations depicting brittle and ductile fracture of solid objects, addressing many of the shortcomings of the earlier approaches, most notably the need to resolve the location and orientation of fracture planes. While these approaches used volumetric meshes, we are motivated to use surface meshes for several reasons: first, the thin geometry of shells requires volume elements with very large aspect ratios commonly leading to poor conditioning of the resulting numerical systems [COS00]; second, implementation (*e.g.*, changing mesh connectivity during fracture) is easier for surface meshes; third, in graphics the vast majority of geometric models are represented as (or easily converted to) triangle meshes.

Related work on plasticity. Some recent graphics work on 3D plastic deformation includes [OBH02] which modeled ductile fracture, [CLH96] which modeled soil, and [DC03] which modeled clay; for 2D deformation, [GDHS03] includes a simple model of plasticity by way of updating the rest shape configuration.

Overview Our paper is structured as follows. First, we derive continuous membrane and bending strains for shells, and introduce their discrete analogues (Sec.). These discrete measures serve as building blocks for discrete models of shell elasticity, plasticity and fracture (Sec.). A robust implementation of our model requires consideration of timestep adjustment criteria and fracture-aware collision detection (Sec.). We demonstrate our implementation on several examples (Sec.).

Membrane and Bending Strains

The key result of this section is to characterize the local deformation of a thin shell as the sum of membrane and bending strains (preview Figure 3). We derive simple discrete expressions for these strains, which serve as a foun-

dation for our discretizations of elasticity, plasticity and the fracture of shells.

Continuous membrane and bending strains.

A geometry of a shell with (local) thickness h is typically represented by a *middle surface* and its extrusion by $h/2$ in both the positive and negative normal directions (see Figure 2). We assume that thickness h is much smaller than the minimal radius of curvature of the middle surface. We consider shells in *undeformed* and *deformed* states, using the convention that quantities accented with tilde (*e.g.*, $\tilde{x}$ vs. x) refer to the deformed state.

Assumptions. We make the common assumptions that (i) the normal lines to the middle surface in the undeformed state are deformed into normal lines to the middle surface in the deformed state (the *Kirchhoff-Love* assumption), and (ii) the distances along the normal lines are preserved (the *normal inextensibility* assumption). Furthermore, we assume that (iii) strains are small. Note that small strains do *not* imply small deformations; for example, a felt hat can be bent drastically without severely stretching the material.

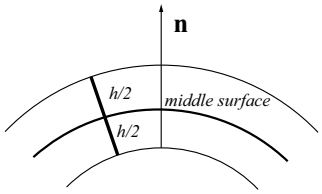

Figure 2: The shell is represented by its *middle surface*. We use x and $\tilde{x}$ to denote quantities related to the deformed and undeformed surface configurations, respectively. $\mathbf{n}$ denotes the unit normal to the surface.

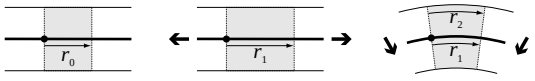

Figure 3: We express the planar strain, $E(z)$, at an offset, z, as the sum of a membrane and bending term. *(left)* Consider the tangent vector, r_0, to a point on the undeformed middle-surface. *(middle)* First, we introduce a membrane strain, E_m, resulting in a deformed tangent, $r_1 = (I + E_m)r_0$. *(right)* Next, we bend the surface, composing the transformation with bending strain and resulting in a deformed tangent, $r_2 = (I + E_c)r_1 \approx (I + E_m + E_c)r_0$, where the approximate equality is valid for small strains.

Strain. Recall that for a map ψ mapping an undeformed configuration of an object to a deformed configuration, the strain measures the change in the squared distance between close points in deformed and undeformed configurations. A strain is a tensor, formally defined as $E = (1/2)(\nabla\psi^T\nabla\psi - I)$; this map can be three-dimensional ("strain") or two-dimensional ("planar strain"). Consider the distance between any two close points $\mathbf{x}$ and $\mathbf{x} + d\mathbf{x}$: strain measures the change in squared distance caused by the deformation, *i.e.*, $d\mathbf{x}^T E d\mathbf{x}$ is $(\psi(\mathbf{x} + d\mathbf{x}) - \psi(\mathbf{x}))^2 - d\mathbf{x}^2$. For thin shells it is assumed that normal strain can be neglected—the components of strain are purely tangential—thus the planar strain completely describes the deformation at any point of the shell. Strain is a purely geometric concept, unrelated to any assumptions about the physics of the materials. However many physical laws describing elasticity, plasticity and fracture are expressed in terms of strains.

Expressions for planar strain. Let $\mathbf{f}(\mathbf{x})$ be the middle surface parameterized by the tangent plane coordinates $\mathbf{x}$. Locally we can regard a shell as the image of a 3D parametric domain $\Omega = \omega \times [-h/2, h/2]$, where ω is a domain in the tangent plane of the middle surface. The maps defining the undeformed and deformed shell are respectively given by

$$\begin{aligned} \Phi : \Omega \to \mathbf{R}^3 \quad &: \quad \Phi(\mathbf{x}, z) = \mathbf{f}(\mathbf{x}) + z\widehat{\mathbf{n}}(\mathbf{x}) \ , \\ \tilde{\Phi} : \tilde{\Omega} \to \mathbf{R}^3 \quad &: \quad \tilde{\Phi}(\tilde{\mathbf{x}}, z) = \tilde{\mathbf{f}}(\tilde{\mathbf{x}}) + z\tilde{\widehat{\mathbf{n}}}(\tilde{\mathbf{x}}) \ . \end{aligned}$$

Because of the inextensibility assumption, the correspondence between material points is given by a map $A : \Omega \to \tilde{\Omega}$ given by $A(\mathbf{x}, z) = (\tilde{\mathbf{x}}(\mathbf{x}), z)$ where $\tilde{\mathbf{x}}(\mathbf{x})$ is the map between tangent planes induced by the correspondence of middle-surface material points. Then the deformation of the shell can be decomposed as $\tilde{\Phi} \circ A \circ \Phi^{-1}$. The transformation between differential volumes is given by the differential of this map at $(\mathbf{x} = 0, z)$. The differential can be computed as a product of three linear 3D operators: $\nabla_{(\tilde{\mathbf{x}},z)}\tilde{\Phi}\nabla A_{(\mathbf{x},z)}(\nabla_{(\mathbf{x},z)}\Phi)^{-1}$, where $\nabla_{(\tilde{\mathbf{x}},z)}$ is the gradient with respect to three variables (x_1, x_2, z). By definition, A has block diagonal form, with a 2x2 block corresponding to planar variables, and 1 corresponding to the normal coordinate z. The 2x2 block for planar variables is $S = \nabla_{\mathbf{x}}\tilde{\mathbf{x}}$, *i.e.*, the gradient of the deformation of the middle surface.

Using the definition of the second fundamental form expressed in tangent plane coordinates $\Lambda_{ij} = (\tau_i \cdot \partial_{x_j}\widehat{\mathbf{n}})(0)$, where τ_i, $i = 1, 2$ are coordinate vectors, we can easily compute

$$(\nabla_{\mathbf{x},z}\Phi)(0) = \begin{pmatrix} I + z\Lambda & 0 \\ 0 & 1 \end{pmatrix} .$$

We note that the second fundamental form in tangent plane coordinates represents the shape operator, *i.e.*, the operator Λ such that for any tangent vector τ $\Lambda\tau$ is the derivative of the unit normal in the direction of τ. We observe that it has block diagonal form with a 2x2 block for planar variables, just as A. This shows that the deformation maps the plane parallel to the tangent plane of the middle surface to itself, and is identity in the normal direction. The planar transformation is computed as the product of three planar map gradients and is

$$F(z) = (I + z\tilde{\Lambda})S(I + z\Lambda)^{-1} . \quad (1)$$

This expression allows us to compute the planar strain at any point along the normal to the middle surface as follows. Expanding the last term in $F(z)$ up to first order in $z\kappa$ (recall that the eigenvalues of Λ are curvatures in the chosen tangent plane basis), and dropping terms proportional to z^2, we obtain

$$F(z) \approx (I + z\tilde{\Lambda})S(I - z\Lambda) \approx S + z(\tilde{\Lambda}S - S\Lambda) .$$

We are interested in planar strain $E = (1/2)(F^T F - I)$, which can be approximated by $(1/2)(F + F^T) - I$. Using $I + E_m \approx (1/2)(S + S^T)$, and dropping second-order terms which are products of membrane strains and terms of order κh we obtain the following expression for E:

$$\boxed{E(z) = E_m + z\Delta\Lambda = E_m + zE_c .}$$

The planar strain is the sum of the middle surface *membrane* strain and a *bending* term proportional to the difference of second fundamental forms expressed in tangent plane coordinates (see Figure 3). We call the second term the *bending strain*, and its subscript refers to curvature.

Discretization of Strains

As we will see in the next section, the continuum models for elasticity, plasticity and fracture can be expressed in terms of membrane and elastic strains defined above. Thus, we can obtain discretizations of these models automatically if we have discretizations for these strains.

In our discretization both strains are assumed to be piecewise constant over triangles of a mesh approximating the shell surface. For the membrane strain, we rewrite a well-known discretization in terms of only mesh invariants. For the bending strain, we introduce a novel discretization with a number of features essential for modeling stiff shells. Both expressions for strain are designed to be closely compatible thus leading to a straightforward implementation.

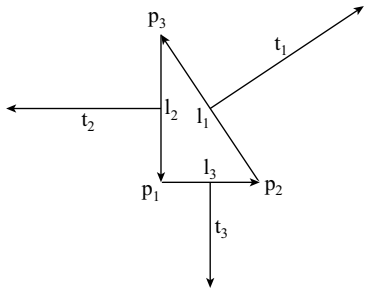

Figure 4: Vectors and points used in the membrane and bending strain calculations. The area of the triangle is $A = (1/2)|n|$.

Membrane strain. Computation of the planar strain on the triangle is well-known and standard both in engineering and graphics literature. As the map from the undeformed to the deformed triangle is affine, the strain is directly determined by the matrix of the gradient of this map: $E_m = 1/2(S^T S - I)$. However, it is convenient to express it in the form in which the dependence on the changes of edge length is explicit. For this purpose we introduce vectors $\mathbf{t}_i$ in the triangle plane, which are perpendicular to the undeformed edges of the triangle, and have the same length as corresponding edges. The vectors pointing counterclockwise along the edges are denoted $\mathbf{v}_i$, $|\mathbf{v}_i| = l_i$, $i = 1,2,3$ (Figure 4).

We observe that by definition of strain, the edge lengths satisfy

$$(\tilde{l}_i^2 - l_i^2) = \mathbf{v}_i^T E_m \mathbf{v}_i \quad (2)$$

where it does not matter in which coordinate frame the tensors and the vectors are expressed. We adopt the notation $s_i = \tilde{l}_i^2 - l_i^2$, and $(\cdot \otimes \cdot)$ for the outer product of two vectors. It turns out to be convenient to use the tensor basis $\mathbf{t}_i \otimes \mathbf{t}_j$, where (i, j, k) is a circular permutation of $(1,2,3)$. A straightforward calculation taking advantage of orthogonality of $\mathbf{v}_j$ and $\mathbf{t}_j$ yields the following expression for the membrane strain:

$$\boxed{E_m = \frac{1}{8A^2} \sum_i s_i \left(\mathbf{t}_j \otimes \mathbf{t}_k + \mathbf{t}_k \otimes \mathbf{t}_j\right).}$$

All factors except s_i depend only on the undeformed state, hence initially we precompute the outer products and areas, and at every timestep we reevaluate only s_i, the deformation-induced change in squared edge-lengths.

Bending strain. To compute the bending strain we need to choose a shape operator discretization. There are many possible ways to define curvature on a polygonal mesh, all of which, under certain assumptions, would converge to continuous curvatures.

One common approach is to consider curvature concentrated at edges, with principal curvatures 0 and $\varphi(\theta)$, where θ is the angle between the normals of the two triangles joined at the edge and $\varphi(\cdot)$ is a suitably chosen function, monotonic in θ. The principal curvature direction corresponding to zero curvature is assumed to be aligned with the edge.

The shape operator corresponding to this case is easily computed, as it has a single nonzero eigenvalue $\varphi(\theta)$, and, therefore, can be expressed as $\varphi(\theta)(\mathbf{t}\otimes\mathbf{t})/l_i$ where $\mathbf{t}$ is the unit vector perpendicular to the average normal of the adjacent triangles and the edge, and l is the edge length. However, this approach suffers from an important problem: the operator does not converge pointwise to the shape operator of the smooth surface as the triangulation is refined. Indeed, the principal curvature directions for this operator do not depend on the principal curvature directions of the sampled smooth surface.

It was proved, however, that the average of this operator over a sufficiently large area converges to the integral of shape operator as the triangulation is refined [CSM03]. In particular, averaging over vertex neighborhoods was used in [ACSD*03, HP04]. We consider an even simpler approach, which matches well our chosen discretization of strain, that is, the averages on triangles. Following the ideas in [HP04], we define the triangle shape operators using projections of the edges to the plane perpendicular to the normal. In the case of triangle the normal is well defined, and the resulting expression is remarkably simple:

$$E_c = \tilde{\Lambda} - \Lambda = \sum_i \frac{(\Delta\varphi_i)}{2Al_i}\mathbf{t}_i \otimes \mathbf{t}_i \ .$$

where $\Delta\varphi_i = \varphi(\tilde{\theta}_i) - \varphi(\theta_i)$. Here the factor $1/2$ accounts for the fact that each edge is shared by two triangles, and the factor $1/A$ takes into account the area over which the operator is averaged. Once more, all factors except φ_i depend only on the undeformed state. Hence after precomputing those factors, at every timestep we reevaluate only φ_i. Contrast this with vertex-based shape operators, which yield a substantially more complex expressions. At the same time, the triangle-centric approach has the needed flexibility: it reproduces all curvature directions (see Figure 5), just as the vertex-centric approach.

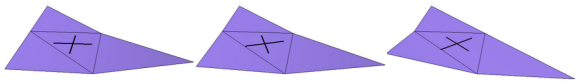

Figure 5: The novel bending strain captures a continuous range of bending directions. Shown are frames from the accompanying video: bending direction changes smoothly as dihedral angles are varied.

The remaining question is the choice of $\varphi(\theta)$, restricted by the following considerations: for θ close to zero the function should behave as θ to obtain convergence to curvature for finer discretizations, and should monotonically increase with θ. [CSM03] uses simply θ, whereas [HP04] uses $2\sin(\theta/2)$. We found the choice $2\tan(\theta/2)$ most suitable as for large θ it results in arbitrarily high bending strain. Our choice is motivated by the observation that during crumpling deformations of real shells almost all of the internal energy is distributed in the neighborhood of ridge and cone formations [Woo02]: using $2\tan(\theta/2)$ (as opposed to $2\sin(\theta/2)$ or θ) the fraction of total energy away from ridge-like edges vanishes with increasing edge sharpness.

Physical Models and Their Discretizations

In this section we consider continuum models for elasticity, plasticity and fracture; all these models are defined and discretized using membrane and bending strains discussed in the previous section.

Elasticity

Conceptually, it is convenient to describe internal elastic shell behavior using elastic energy. As the geometric deformation of the shell at a given point is completely described by the pair of strains E_m and E_c, any such energy is a function of the invariants of these tensors. Our discretization is based on the idea of representing these tensors using invariant quantities (edge lengths and dihedral angles). This ensures that invariant expressions for any energy expressed in terms of membrane and bending strains can be obtained by simple substitution of the discrete expressions for strains.

We use energies that can be separated into two parts, one depending only on the membrane strain and the other

only on the bending strain. The membrane surface energy density is well-known and standard:

$$W_{membrane} = h\frac{Y}{2(1-\nu^2)}\left((1-\nu)\mathrm{Tr}(E^2)+\nu(\mathrm{Tr}E)^2\right)$$

The coefficients Y and ν are the Young modulus and the Poisson ratio for the material. We consider two examples of bending energy. [GDHS03] uses energy density $C\Delta H^2$ where ΔH^2 is the change in the mean curvature. The use of that energy is based on purely geometric considerations. (In the case of a flat undeformed configuration it corresponds to the Willmore energy.) We observe that change of the mean curvature is simply the trace of our bending strain $\Delta H = \mathrm{Tr}E_c$, as for our choice of coordinates the shape operators Λ_i have curvatures as eigenvalues. Thus the "Discrete Shells" bending energy density is simply

$$W^{DS}_{bending} = C(\mathrm{Tr}E_c)^2$$

A more complex model derived from physics considerations is Koiter's shell model [Koi66]. It is obtained by using the zero normal stress assumption to convert 3D strains and stresses to two dimensions and a linear elasticity constitutive law to derive the equation for energy. Using our bending strain, this model yields an expression for bending energy density which is remarkably similar to the membrane energy density:

$$W^{Koiter}_{bending} = \frac{Yh^3}{24(1-\nu^2)}\left((1-\nu)\mathrm{Tr}(E_c^2)+\nu(\mathrm{Tr}E_c)^2\right)$$

While this elastic energy is not the focus of our work here, we have provided it for completeness, and a detailed derivation will follow in a technical report. The total elastic energy in both cases is $W_{membrane}+W_{bending}$. The complete equations of motion for the discretized shell are obtained by differentiating the discrete elastic energy expressions with respect to vertex positions to obtain forces and adding external and damping forces.

Observe that the elastic energy density we get using our strain discretizations is just a quadratic function of changes of squared lengths, s_i, and functions of dihedral angles, $\Delta\varphi(\theta)_i$. Thus the complexity of this energy is close to that of ad hoc edge-length and angle-based energies used in cloth simulation.

Temporal discretization using the Newmark scheme. We integrate the system forward in time using the Newmark scheme [WKMO00]:

$$\begin{aligned}\mathbf{x}_{i+1} &= \mathbf{x}_i+\Delta t_i\dot{\mathbf{x}}_i+\Delta t_i^2\left((1/2-\beta)\ddot{\mathbf{x}}_i+\beta\ddot{\mathbf{x}}_{i+1}\right),\\ \dot{\mathbf{x}}_{i+1} &= \dot{\mathbf{x}}_i+\Delta t_i\left((1-\gamma)\ddot{\mathbf{x}}_i+\gamma\ddot{\mathbf{x}}_{i+1}\right),\end{aligned}$$

where Δt_i is the duration of the i^{th} timestep, and $\dot{\mathbf{x}}_i$ and $\ddot{\mathbf{x}}_i$ are the state velocity and acceleration at the beginning of the i^{th} timestep, respectively. The adjustable parameters β and γ are linked to the accuracy and stability of the time scheme. Newmark is either an explicit ($\beta = 0$) or implicit ($\beta > 0$) integrator: we used the standard $\beta = 1/4$ for final production, and $\beta = 0$ to aid in debugging. Newmark gives control over numerical damping via its second parameter γ, which is discussed in West *et al.* [WKMO00].

Plasticity

Plasticity model for solids. Our plasticity formulation for shells is based on the same 3D plasticity model as used in [OBH02].

We start with briefly explaining the plasticity model for solids; more details can be found in many mechanics texts. We found the general description in [HR99] useful for derivation of the shell plasticity model. Typically, plasticity models are defined in terms of stress and strain; by assuming a linear elastic law, we describe plasticity in terms of strain only.

The plasticity model that we use relies on the following basic assumptions; these are easiest to understand in the one-dimensional case, where the strain ε is a scalar. The state of a plastic object is characterized by *plastic strain*, p, which is the strain that remains in the absence of external (including body) forces. The simplest plastic behavior is ideal plasticity: when the total strain, ε, exceeds a threshold ε_0, all further increase in the total strain is converted to plastic strain. This model is not very realistic: most materials have some form of *hardening*, *i.e.*, the plastic threshold depends on the accumulated plastic strain p. A simple model for this effect is to assume that plastic regime starts when the *elastic strain* $\varepsilon_e = \varepsilon - p$ reaches $\varepsilon_0 + \gamma p$, *i.e.*, the threshold grows linearly with the plastic strain. This model for hardening is called *linear kinematic hardening*. The equation for the change in plastic strain is immediately obtained from $\varepsilon - p = \varepsilon_0 + \gamma p$: $\Delta p = \Delta\varepsilon/(1+\gamma)$. The analogues of this formula is usually referred to as *plastic update*.

The plastic update formula immediately shows how plasticity can be implemented computationally. First, compute the increment in total strain keeping plasticity fixed; then update plastic strain using the plastic update formula.

In 3D strain is no longer a scalar; we use a more general yield function $\varphi(\varepsilon_e, \mathbf{p})$ defined for tensor strains to determine the transition to the plastic regime and computing the plastic update (the one-dimensional analogue is $\varepsilon - \gamma p - \varepsilon_0$). Derivation of such functions is based on

physical observations; in particular, it was observed that plastic deformation is primarily created by anisotropic strain; for this reason, in the plasticity model that we use the yield function depends only on the traceless part of the strain, *i.e.*, $\varepsilon_e^D = \varepsilon_e - \frac{1}{3}\mathrm{Tr}\varepsilon$. The *von Mises yield function* for ideal plasticity is simply the Frobenius norm of ε_e^D, *i.e.*, $\sqrt{\sum_{ij}(\varepsilon_{ij}^D)^2}$ minus a plastic threshold. If hardening is present, then, similar to the one-dimensional case, only a part of anisotropic stress is used in the yield function: $\varphi(\varepsilon_e, \mathbf{p}) = \|\varepsilon_e^D - \gamma\mathbf{p}\| - \varepsilon_0$. The constant γ in this formulation is related to the standard kinematic hardening coefficient h_{kin} as $\gamma = h_{kin}/2\mu$ where μ is the Lamé constant for the material.

Once the yield function becomes positive, we need to define how the plastic strain is increased to return it back to zero. Unlike the one-dimensional case, the linear elasticity law and the condition $\varphi(\varepsilon_e, \mathbf{p}) = 0$ are insufficient to determine the plastic update uniquely. Additional physical considerations (the principle of maximum plastic work) result in the update formula of the form

$$\Delta\mathbf{p} = \lambda(\varepsilon_e^D - \gamma\mathbf{p}),$$

where $\lambda = \varphi(\varepsilon_e + \Delta\varepsilon))/(1+\gamma)$.

Shell plasticity. Just as it is the case for elasticity models, the plasticity model for shells can be derived from the model for solids. An additional assumption that we make is that for a given point on the middle surface all material points along the normal are simultaneously in plastic or elastic mode, with the yield function value obtained by averaging the values along the normal.

We also use the planar stress assumption to convert 3D strains and stresses to 2D. Once these assumptions are made can obtain explicit formulas for plastic update by routine calculations.

All 2D strain quantities at a point of the shell are characterized using a pair of 2D tensors (A_m, A_c), such that the strain at distance z from the middle surface has the form $A_m + zA_c$; index m refers to the middle surface quantity, c refers to the term associated with surface curvature, as it was done for the elastic strain. In particular, we use the total strain $E = (E_m, E_c)$, plastic strain $P = (P_m, P_c)$, elastic strain $E^e = E - P$. For convenience we define a constant $c = (1/6)(1+\nu)/(1-\nu)$. The expression for the yield function and plastic updates are best expressed using an auxiliary strain A:

$$A_i = c\mathrm{Tr}E_i I + E_i - (1+\gamma)P, \quad i = m, c.$$

Using this strain, we can express the yield function Φ as follows:

$$\Phi(E,P) = F(E,P) - \varepsilon_0 \ ,$$
$$F(E,P) = \frac{h}{2}\left(\mathrm{Tr}A_m^2 + (\mathrm{Tr}A_m)^2\right) + \frac{h^3}{24}\left(\mathrm{Tr}A_c^2 + (\mathrm{Tr}A_c)^2\right) \ .$$

We have introduced the function $F(E,P)$, which in the 3d yield function corresponds to the norm of the traceless stress $\|\varepsilon_e^D - \gamma\mathbf{p}\|$. The update formulas for 2D strain components are

$$\Delta P_i = \frac{\Phi(E,P)}{1+\gamma} F(E,F) A_i \ . \tag{3}$$

Discretization. The only additional variable needed to describe plastic materials with linear kinematic hardening is plastic strain. In our discretization, we represented it exactly in the same way elastic strain is represented, *i.e.*, using per-triangle variables. For the membrane component, we store a triple of squared edge length changes, and for the bending component, a triple of angle changes. The plastic update is performed explicitly at each time step: we apply the continuous formulas for the yield function and plastic update directly to the discrete strains.

Fracture

We model the fracture of a shell when in a small region the magnitude of *principal strain* exceeds a material-specific tensile or compressive threshold. Here principal strain refers to an eigenvalue of the strain tensor, $E(z)$; the associated eigenvector gives the *principal strain direction*. The material fractures along the *fracture line* perpendicular to the principal strain direction. O'Brien [OH99] described a similar model of fracture (with a different discretization), and explained that while the resulting behavior is not truly physical, as the important plastic region near the tip of a crack is not correctly modeled, it is adequate for animating fracture of solid objects.

One can observe[1] that the *principal strains*, *i.e.*, the eigenvalues $|\lambda_1(z)| > |\lambda_2(z)|$ of $E(z)$, take extremal values for $z = \pm h/2$. In our model, the material instantaneously fails at every point where $|\lambda_1(+h/2)| > \lambda_\mathrm{f}$ (respectively $|\lambda_1(-h/2)| > \lambda_\mathrm{f}$). Note that it would be very easy to assign different thresholds to tensile ($\lambda_1(+h/2) > \lambda_{\mathrm{f}+} > 0$) or compressive ($\lambda_1(+h/2) < \lambda_{\mathrm{f}-} < 0$) strains.

[1] The principal strain λ_i can be expanded in the form $\lambda_i^m + z(\lambda_1^c(1+\cos 2\beta) + \lambda_1^c(1-\cos 2\beta))/2$ up to second order in z, where λ_i^m are the eigenvalues of E_m, λ_i^c are the eigenvalues of E_c, and β is the angle between the dominant eigenvectors of E_m and E_c.

Discretization of fracture. We simulate fracture events *per face*: for every face, we compute eigenvalues of $E(h/2)$ and $E(-h/2)$, and if any exceed λ_f then then the face is deemed fractured and is split as shown in Figure 6. Note that in general a face fractures due to a combination of membrane and bending strains. Note that unlike earlier approaches, we allow simultaneous failures at different points in the surface, *i.e.*, we do not impose an arbitrary temporal ordering on fracture events. This improvement is possible due to fracture-aware timestepping code. As we will see in Section , a key consequence is that, unlike in earlier work on simulating fracture, we do not need to model the propagation of excess fracture energy to nearby regions of material.

We have also experimented with plastic behavior near the crack tip and have found this to be a good way to dissipate excess energy during fracture (in Section we treat plasticity). Immediately after a face is split into two, we can absorb some percentage of the elastic strain into the local plastic state. As expected, fractures propagate further if less strain is absorbed plastically. While this technique is effective and provides a simple and elegant alternative to damping, the question of modeling behavior at the crack tip remains open.

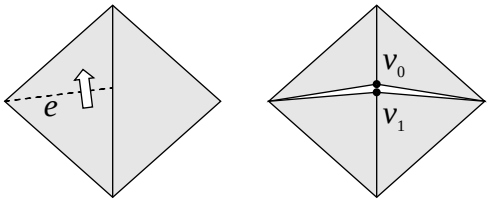

Figure 6: *(left)* We discretize fracture events by examining each mesh face in isolation, splitting those faces with excessive strain along the unique fracture line e that is perpendicular to the direction of greatest strain and incident on a face vertex. *(right)* Splitting along e introduces v_0 and v_1. If the introduced vertices are interior (away from surface boundary), then the adjacent face is also split. This secondary split does not introduce new vertices nor further splits.

Vertex budging improves mesh quality. The potential to capture a continuous range of fracture orientations is a mixed blessing. It allows for smooth, "antialiased," fracture boundaries, but it may demand that we create arbitrarily thin "sliver" triangles. When a fracture line is nearly the same as an existing mesh edge, we locally reparameterize the surface by sliding the existing edge onto the desired fracture line; we call this operation a *vertex budge*

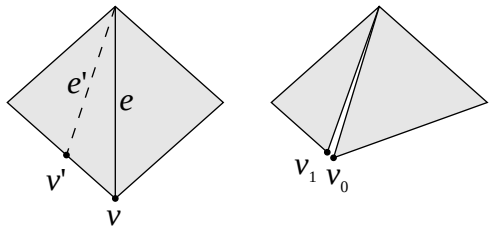

Figure 7: Arbitrarily-oriented fractures can potentially create arbitrarily-thin faces. To prevent this, we locally reparameterize of the surface via *vertex budging*. *(left)* If a proposed face split will introduce a vertex v' near an existing vertex v (for clarity we have exaggerated the distance between the two), we instead plastically deform v to the position of v', thus aligning an existing mesh edge e with the desired fracture line e'. Note that budging does not alter strains. *(right)* We can effect fracture by splitting along e.

because it amounts to repositioning a vertex in parametric space (see Figure 7). We regard vertex budge as a change in discretization of a smooth surface we approximate. This implies that vertex budging should not modify the strains on the mesh, even in the vicinity of the budge. To implement a vertex budge, we measure the change in strain, over each face, cause by the repositioning, and we absorb the change plastically (Section).

The combination of arbitrary fracture orientations and vertex budging allows our final animations to have smooth fracture boundaries, where they should, and more generally to appear as if the coarse-mesh simulations were carried out on a fine mesh (see Figure 8). Surprisingly, we found that budging produced better results than increasing mesh resolution, thus promising to be a great source for reducing computational cost. We feel that budging should be further explored in other physical-simulation settings.

Implementation

Several considerations lead to a more robust implementation:

Simulation Loop

Our simulation loop moves the simulation forward in time while ensuring that important events are not missed. The most important events to resolve are fractures and sudden large forces. For a particular step forward in time, we check a list of criteria and if any criteria are not satisfied

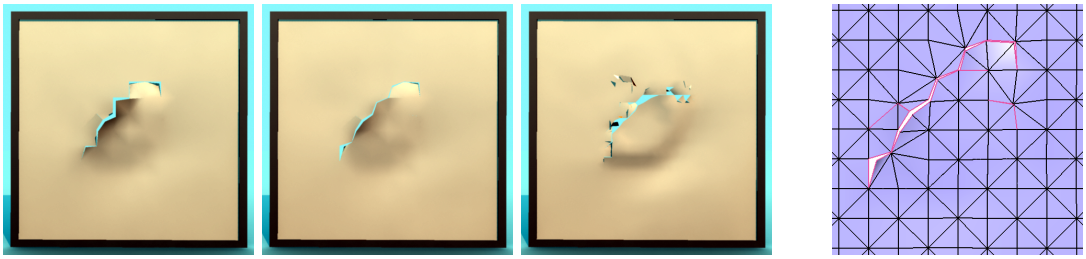

Figure 8: Budging drastically improves the quality of fracture edges; compare identical simulations *(left)* without budging and *(middle)* with budging. Furthermore, budging produced better torn fringes than running the simulation with *(right)* four times as many faces. The action of budging can be seen on the detailed mesh *(far right)*.

then we reduce the time step and try again. This dynamic search is controlled by the current *search level* k, which dictates an effective time step of $2^{-k}\Delta t$. The search level is controlled by the **SimulationLoop** algorithm:

SimulationLoop

```
1   k := 0
2   While t < t_end
3     t' := t + 2^-k Δt // proposed (level-k) timestep
4     compute proposed state at t' // Sec.
5     if k > k_max or all criteria satisfied // Sec.
6       t := t' // successful timestep
7       accept proposed state
8       update plastic state // Section
9       fracture over-strained faces // Section
10      k := max(k − 1, 0) // pop search level
11    else
12      discard proposed state
13      k := k + 1 // push search level
```

All simulations for this paper were run using a maximum search level $k_{\max}$ of ten, representing a search resolution of roughly $1/1000^{\text{th}}$ of the current time step.

Timestep Adjustment Criteria

Limiting force rate. We limit the rate of change of forces on the object over a time step so that objects can respond naturally to large forces. For example, if we drop an egg onto a floor it will fall freely until entering the floor's proximity region. If the time steps are large then at time $t+\Delta t$ some area of the egg will have penetrated deeply into the floor's proximity region and received a corresponding large force. This large force is an artifact of the time step and the resulting behavior will disagree with the real behavior. The most obvious result of overly-large forces is an over-abundance of fracturing.

We compute the magnitude of the maximum force of the proposed state, $F_{t+\Delta t}$, and compare it to the same quantity of the current state. All simulations in this paper limit the relative value $r = (F_{t+\Delta t} - F_t)/F_t$ to 1.1, or a %10 increase over the current state. In this way we limit drastic changes to the state of the simulation and allow reactions to proceed naturally. Note that since we dynamically change the search level k the time step is automatically reduced for temporary events such as a floor bounce. The search level decreases and the effective time step returns to its previous size after the magnitudes of the forces have diminished.

Resolving fracture events. In addition to large forces, we also resolve fracture events. In essence, overstepping a fracture event is an attempt to integrate over a sharp discontinuity. Such an attempt leads to over-strained elements which experience large forces, and consequent undesirable artifacts, in particular extraneous fractures (see Figure 9). Rather than overstepping and then using a heuristic model for propagating excess strain, we discard the proposed state. Fracturing at the correct time allows nearby faces to relax with respect to the new boundary.

Fracture-aware collision detection and response

Responding to collisions is a key ingredient for realistic simulation. The case of surfaces is more difficult because, unlike solids, surfaces do not have an inside/outside orientation; Bridson *et al.* [BFA02] and Baraff *et al.* [BWK03] recently presented robust approaches to deal with collisions in the surface setting. To deal with the lack of orientation, we never accept a state in which there are intersections, and we use the popular penalty force approach to

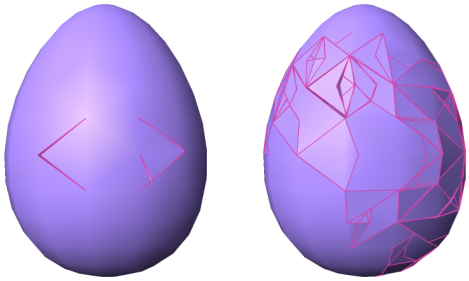

Figure 9: Searching in time for important simulation events can eliminate undesirable artifacts. We demonstrate this with a simulation of a dropped egg shell hit by a very small force. *(left)* Searching in time creates only the expected fractures; *(right)* in contrast, disabling the search routine leads to large numbers of unexpected fractures and brittle behavior.

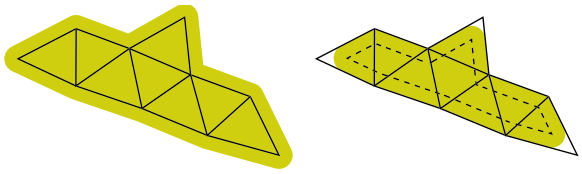

Figure 10: To accommodate fracture, the repulsion field is inset along mesh boundaries. The shaded region represents the repulsion field of a piece of mesh *(left)* before insetting and *(right)* afterward: dashed lines indicate the inset boundary; note that the repulsion field now lies entirely within the area of the mesh.

prevent intersections [MW88]. Our contribution is a treatment of fracture events which (a) alter mesh connectivity, and (b) introduce abutting disconnected geometry.

Detection. For collision detection, we rely on a hierarchy of axis-aligned bounding boxes to cull the number of pairwise triangle-intersection tests. The hierarchy is initially constructed top-down. At every timestep the extents of the bounding boxes are update bottom-up. The hierarchy structure is *incrementally* updated after fracture events, since they *locally* modify mesh connectivity. Furthermore, every constant (*e.g.*, 60) number of frames, the hierarchy structure is rebuilt from scratch to ensure temporal-spatial coherence. More sophisticated culling might be achieved by adapting curvature-based culling to our setting [VCM95].

Response. Our approach to collision response builds on the popular penalty force approach. We surround the triangle mesh by a repulsive force field of thickness $h/2$. We consider all vertex-face and edge-edge interactions between the triangles, applying equal and opposite repulsive forces to interacting pairs. When fractures are created, disconnected components are instantaneously coincident: if not dealt with, the standard collision response would inappropriately generate repulsive forces. Indeed, when the material fractures, the collision code should not prevent the abutting pieces from remaining where they are, but it should prevent them from penetrating deeply. To that end, we *inset* the repulsion field around boundary edges (see Figure 10). The inset field allows the recently-disconnected abutting geometry to not interact unless it begins to penetrate. The drawback to the penalty-force approach is that inset edges may penetrate by a small distance ($O(h)$); we have not found this to be a problem. An alternative approach might be to use analytic constraints.

Results

We simulated the shattering of a glass lightbulb when hit by a fast-moving projectile (see Figure 1). Observe the high variance in the size of the shards, a typical characteristic of glass materials. Note that during rendering we added a slight thickness to the material. The simulation took approximately 70min on a 2.4GHz P4 Xeon, including collision detection.

We simulated the puncturing of sheets with varying fracture thresholds (Figure 11), and with varying plastic parameters. As a comparison, we also simulated a sheet without the budging operation (Figure 8). Observe that that we have intentionally omitted extraneous damping from our simulation. These runs required between 8min to 30min of computation time depending on processor, the number of faces created during fracture, and the complexity of collisions.

Although plasticity is vital for simulating a wide range of fracturing materials, it also stands on its own, as we demonstrate with a simulation of a dropped, then dented, tube (see Figure 12). In the accompanying animation, observe that as the tube squishes, some of the deformation is stored as elastic energy resulting in a bounce, while the rest becomes plastic work resulting in a dent. The entire computation took 11min on a P4 1.7GHz processor. The accompanying video also contains animations of a projectile shattering a diving board and a bowl; again plastic and fracture coefficients are varied to demonstrate a range of materials.

Figure 11: We simulate the puncture of an elastic sheet by a fast projectile. By varying the fracture threshold, λ_f, we obtain different behaviors. From left to right, $\lambda_f = 0.02, 0.001, 0.0001$.

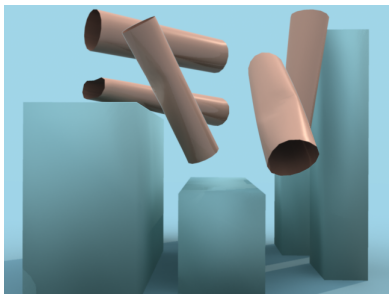

Figure 12: We model plasticity with kinematic hardening, which gives us a range of inelastic materials. We simulate an elastoplastic tube as it falls, dents, and bounces off boxes.

Conclusion. We derived a novel formulation for the strain of a thin shell in terms of membrane and bending components, and we presented a simple and elegant discretization of the strain expressed in terms of the surface invariants of a triangle mesh. This per-triangle strain measure serves as a unifying foundation for our models of fracture and plasticity. Our current shell model does not capture change in thickness due to deformation. Modeling the thinning of an object due to elastoplastic stretching or bending may help to capture the effects of an object weakening due to strain, and we plan to extend our model in this direction.

References

[ACSD*03] ALLIEZ P., COHEN-STEINER D., DEVILLERS O., LÉVY B., DESBRUN M.: Anisotropic polygonal remeshing. *ACM Transactions on Graphics 22*, 3 (July 2003), 485–493.

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

[BWK03] BARAFF D., WITKIN A., KASS M.: Untangling cloth. *ACM Transactions on Graphics 22*, 3 (July 2003), 862–870.

[CLH96] CHANCLOU B., LUCIANI A., HABIBI A.: Physical models of loose soils dynamically marked by a moving object. In *Computer Animation '96* (June 1996), pp. 27–35.

[COPar] CIRAK F., ORTIZ M., PANDOLFI A.: A cohesive approach to thin-shell fracture and fragmentation. *Computer Methods in Applied Mechanics and Engineering* (to appear).

[COS00] CIRAK F., ORTIZ M., SCHRÖDER P.: Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *Internat. J. Numer. Methods Engrg. 47*, 12 (2000), 2039–2072.

[CSM03] COHEN-STEINER D., MORVAN J.-M.: Restricted delaunay triangulations and normal cycle. In *Proc. 19th Annu. ACM Sympos. Comput. Geom.* (2003), pp. 237–246.

[DC03] DEWAELE G., CANI M.-P.: Interactive global and local deformations for virtual clay. In *Pacific Graphics* (2003). Canmore, Canada.

[GDHS03] GRINSPUN E., DESBRUN M., HIRANI A., SCHRÖDER P.: Discrete shells. In *Proceedings of ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2003), Breen D., Lin M., (Eds.).

[Gre73] GREENSPAN D.: *Discrete Models*. Addison-Wesley, 1973.

[HP04] HILDEBRANDT K., POLTHIER K.: Anisotropic filtering of non-linear surface features. In *Proceedings of MINGLE Workshop* (2004).

[HR99] HAN W., REDDY B. D.: *Plasticity*, vol. 9 of *Interdisciplinary Applied Mathematics*. Springer-Verlag, New York, 1999. Mathematical theory and numerical analysis.

[Koi66] KOITER W. T.: On the nonlinear theory of thin elastic shells. I, II, III. *Nederl. Akad. Wetensch. Proc. Ser. B 69* (1966), 1–17, 18–32, 33–54.

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

[MW88] MOORE M., WILHELMS J.: Collision detection and response for computer animation. In *Computer Graphics (Proceedings of SIGGRAPH 88)* (Aug. 1988), vol. 22, pp. 289–298.

[MW01] MARSDEN J. E., WEST M.: Discrete mechanics and variational integrators. *Acta Numerica* (2001), 357–514.

[NF99] NEFF M., FIUME E. L.: A visual model for blast waves and fracture. In *Graphics Interface '99* (June 1999), pp. 193–202.

[NTB*91] NORTON A., TURK G., BACON B., GERTH J., SWEENEY P.: Animation of fracture by physical modeling. *The Visual Computer*, 7 (1991), 210–219.

[OBH02] O'BRIEN J. F., BARGTEIL A. W., HODGINS J. K.: Graphical modeling and animation of ductile fracture. In *AMC Transactions on Graphics* (2002), ACM Press, pp. 291–294.

[OH99] O'BRIEN J. F., HODGINS J. K.: Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH 99* (Aug. 1999), Computer Graphics Proceedings, Annual Conference Series, pp. 137–146.

[SY98] SHEN J., YANG Y.-H.: Deformable object modeling using the time-dependent finite element method. *Graphical Models and Image Processing 60*, 6 (Nov. 1998), 461–487.

[TF88] TERZOPOULOS D., FLEISCHER K.: Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Computer Graphics (Proceedings of SIGGRAPH 88)* (Aug. 1988), vol. 22, pp. 269–278.

[VCM95] VOLINO P., COURCHESNE M., MAGNENAT THALMANN N.: Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proceedings of SIGGRAPH 95* (1995), ACM Press, pp. 137–144.

[WKMO00] WEST M., KANE C., MARSDEN J. E., ORTIZ M.: Variational integrators, the newmark scheme, and dissipative systems. In *International Conference on Differential Equations 1999* (Berlin, 2000), World Scientific, pp. 1009 – 1011.

[Woo02] WOOD J.: Witten's lectures on crumpling. *Physica A: Statistical Mechanics and its Applications 313*, 1–2 (October 2002), 83–109.