Eurographics/SIGGRAPH Symposium on Computer Animation (2003)
D. Breen, M. Lin (Editors)

Discrete Shells

Eitan Grinspun[†] Caltech
Anil N. Hirani Caltech
Mathieu Desbrun USC
Peter Schröder Caltech

Abstract

In this paper we introduce a discrete shell model describing the behavior of thin flexible structures, such as hats, leaves, and aluminum cans, which are characterized by a curved undeformed configuration. Previously such models required complex continuum mechanics formulations and correspondingly complex algorithms. We show that a simple shell model can be derived geometrically for triangle meshes and implemented quickly by modifying a standard cloth simulator. Our technique convincingly simulates a variety of curved objects with materials ranging from paper to metal, as we demonstrate with several examples including a comparison of a real and simulated falling hat.

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

1. Introduction

Thin shells are thin flexible structures with a high ratio of width to thickness (> 100). While their well-known counterparts, thin *plates*, relax to a *flat* shape when unstressed, thin *shells* are characterized by a *curved undeformed configuration*. Cloth, recently studied in the animation literature, may be modeled as a thin plate, since garments are typically constructed from flat textiles. In stark contrast, thin-walled objects which are naturally curved (*e.g.*, leaves, fingernails), or put into that shape through plastic deformation (*e.g.*, hats, cans, carton boxes, pans, car bodies, detergent bottles) are thin shells and *cannot be modeled using plate formulations.*

Thin shells are remarkably difficult to simulate. Because of their degeneracy in one dimension ("thinness"), shells do not admit to straightforward tessellation and treatment as three-dimensional solids[‡]. Robust finite element methods for thin shell equations continue to be an active and challenging research area[2, 10]. In contrast, thin *plate* equations tailored to cloth modeling have seen successful numerical treatment; such approaches cannot account for the *structural rigidity* that arises from a *curved* undeformed configuration. Thin plates have been modeled by mass-spring networks: resistance to bending is effected by "diagonal" springs connected to opposite corners of adjacent mesh faces. These techniques do not carry over to curved undeformed configurations: the diagonal springs are insensitive to the sign of the dihedral angles between faces.

Recently, novel numerical treatments of shells, significantly more robust than earlier approaches, have been introduced in mechanics[9] and graphics[18, 19]. These continuum-based approaches use the Kirchoff-Love constitutive equations, whose energy captures curvature effects in curved coordinate frames; consequently they model a rich variety of materials. The novel approaches remain relatively complex and computationally expensive: shells made of stiff materials are considered *challenging* and *costly* to simulate.

Contribution Building upon the pioneering work of Terzopoulos *et al.*[25] and the beautiful geometric interpretation of shells by Ciarlet[8], we present a simple and realistic technique for simulating thin shells. Our *discrete model* of shells captures the same characteristic behaviors as more complex models, with a surprisingly simple implementation: *a small change to a cloth simulator yields shell simulations at a negligible performance penalty*. We demonstrate the realism of our approach through various examples including comparisons with real world footage (see Figure 1).

2. A Discrete Shell Model

Our model of thin shells is governed by nonlinear *membrane* and *flexural* energies. These energies measure differences between the undeformed configuration $\bar{\Omega}$ and deformed configuration Ω. We take measurements which are invariant under rigid body transformations of the undeformed and/or deformed configurations: this ensures that our internal forces conserve linear and angular momentum. We use an arbitrary 2-manifold triangle mesh to describe the shell geometry, and denote a mesh edge (resp. face) with the letter e (resp. f). Let $\varphi : \bar{\Omega} \mapsto \Omega$ be the piecewise-affine deformation map from the undeformed to the deformed surface, mapping every face

[†] eitan@cs.caltech.edu

[‡] The numerics of such approaches become catastrophically ill-conditioned, destroying convergence and/or accuracy.

(resp. edge, vertex) of the undeformed mesh to the corresponding face (resp. edge, vertex) of the deformed mesh.

Membrane Elastic surfaces resist stretching (local change in area) and shearing (local change in length but *not* area). Feynman[14], then Terzopoulos[25] introduced membrane energies to the graphics community and presented discretizations of this energy. Other models followed[4, 7, 20].

While some materials such as rubber sheets may undergo significant deformations in the stretching or shearing (*membrane*) modes, we focus on inextensible shells which are characterized by mostly *isometric* deformations, i.e., possibly significant deformations in bending but unnoticeable deformation in the membrane modes. Works on cloth simulation similarly focus on inextensible plates[4, 5]. Most membrane models for triangle meshes satisfy this small-membrane-strain assumption with choice of suitably large membrane stiffness coefficients. We are not partial to a particular treatment of membranes: the various models in the literature all serve for our purposes†.

So far we have only discussed energies that measure membrane (*intrinsic*) deformations. However, when a surface bends—an *extrinsic*[17] deformation—*flexural energy* comes into play.

Flexure Models in mechanics are based on invariant measures, *i.e.*, quantities which are not affected by rigid-body transformations of the coordinate frame. Typically, this has led to formulations based on tensors. For example, shell models use the difference of the second fundamental forms[17] in the deformed and undeformed configurations (pulling back the deformed tensor onto the undeformed configuration). The seminal work of Terzopoulos *et al.* introduced such tensorial treatments to the graphics community[25]. These treatments derive tensorial expressions over smooth manifolds, and as a final step discretize to carry out the numerics. In contrast, we define a *discrete* constitutive model by applying geometric operators over piecewise-linear surfaces. In both earlier treatments and our discrete treatment *the underlying geometry is the same.* However, the resulting expressions are simpler in the discrete approach.

The shape operator[17] is the derivative of the Gauss map‡: geometrically, it measures the local curvature at a point on a smooth surface. Our bending energy is an extrinsic measure of the difference between the shape operator evaluated on the deformed and undeformed surfaces. We express this

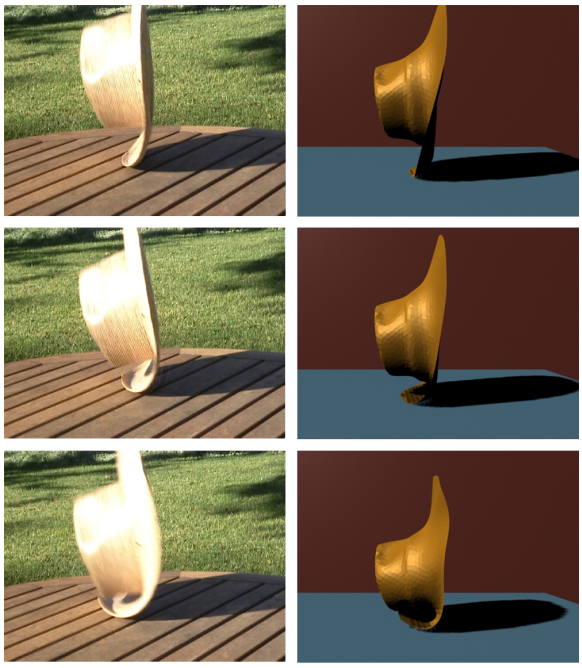
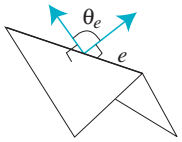


Figure 1: *Real footage vs. Simulation: left, a real hat is dropped on a table; right, our shell simulation captures the bending of the brim. Notice that volumetric-elasticity, plate, or cloth simulations could not capture this behavior, while earlier work on shell simulation required significant implementation and expertise (see also the color plate).*

difference as the *squared difference of mean curvature*:

$$[\mathrm{Tr}(\varphi^* \mathrm{S}) - \mathrm{Tr}(\bar{\mathrm{S}})]^2 = 4(H \circ \varphi - \bar{H})^2 \,, \qquad (1)$$

where $\bar{\mathrm{S}}$ and S are the shape operators evaluated over the undeformed and deformed configurations respectively; likewise $\bar{H}$ and H are the mean curvatures; $\varphi^*\mathrm{S}$ is the pull-back of S onto $\bar{\Omega}$, and we use $\mathrm{Tr}(\varphi^* S) = \varphi^* \mathrm{Tr}(S) = \mathrm{Tr}(S) \circ \varphi = H \circ \varphi$ for a diffeomorphism φ. This measure is *extrinsic*: it sees only changes in the *embedding* of the surface in $\mathbb{R}^3$. This measure is *invariant under rigid-body transformations*: this ensures conservation of linear and angular momentum. Integrating (1) over the reference domain we find the continuous flexural energy $\int_{\bar{\Omega}} 4(H \circ \varphi - \bar{H})^2 d\bar{A}$. Discretizing the integral over the *piecewise linear mesh* that represents the shell (see the appendix for a derivation), we express our *discrete flexural energy* as a summation over mesh edges,



$$W_B(\mathbf{x}) = \sum_e (\theta_e - \bar{\theta}_e)^2 \|\bar{e}\| / \bar{h}_e \,, \qquad (2)$$

where θ_e and $\bar{\theta}_e$ are corresponding complements of the dihedral angle of edge e measured in the deformed and undeformed configuration respectively, and $\bar{h}_e$ is a third of the average of the heights of the two triangles incident to the edge e (see the appendix for another possible definition of $\bar{h}_e$). A contemporaneous development (published in this volume) has arrived at a similar formula: Bridson *et al.*[6] treat the bending and wrin-

† Similarly to Terzopoulos *et al.*[25], our membrane and bending energies are kept separate—see (3). The stiffness coefficients of the membrane (or plate) model directly carry over to the shell model. Knowing the stiffness of an aluminum plate is knowing the stiffness of an aluminum shell.

‡ This is the map from the surface to the unit sphere, mapping each surface point to its unit surface normal.

kling of cloth and give a very compelling demonstration of their model in production footage. Alias|Wavefront, in their Maya software, have implemented a mechanism for specifying non-zero angles for creases in cloth. This suggests a formula also similar to (2), and yet the associated publication[4] explicitly uses a *flat* reference configuration. Creases based on this publication permit at most *developable* reference configurations (via a series of folds, without stretching, these become planar). However, many interesting shells are not developable.

Following the reasoning for (1), we could have formed a second energy term taking the determinant instead of the trace of S. This would lead to a difference of Gaussian curvatures, but this is always zero under isometric deformations (pure bending). This is not surprising, as Gaussian curvature is an *intrinsic* quantity, *i.e.*, it is independent of the embedding of the two-dimensional surface into its ambient three-dimensional space. In contrast, flexural energy measures *extrinsic* deformations.

Dynamics A simple, physically-motivated shell model can thus be expressed by the sum of membrane and flexural energies,

$$W = W_M + k_B W_B \tag{3}$$

where k_B is the bending or flexural stiffness, and W_M is the membrane energy, adopted from the various alternatives in the literature. In our reference implementation, we use $W_M = k_L W_L + k_A W_A$, where $W_L = \sum_e (1 - \|e\|/\|\bar{e}\|)^2 \|\bar{e}\|$, a summation over (lengths of) edges, measures local change in length, while $W_A = \sum_A (1 - \|A\|/\|\bar{A}\|)^2 \|\bar{A}\|$, a summation over (areas of) triangles, measures local change in area. Separation of stretching and bending is not new to graphics[25]. While a complete treatment should consider coupling of membrane and bending modes, for animation this separation is a reasonable simplification. We model different materials, from rubber to aluminum, by tuning the membrane- and bending-stiffness.

Our dynamic system is governed by the ordinary differential equation of motion $\ddot{\mathbf{x}} = -\mathbf{M}^{-1} \nabla W(\mathbf{x})$ where $\mathbf{x}$ is the vector of unknown DOFs (*i.e.*, the vertices of the deformed geometry) and $\mathbf{M}$ is the mass matrix. We use the conventional simplifying hypothesis that the mass distribution is lumped at vertices: the matrix M is then diagonal, and the mass assigned to a vertex is a third of the total area of the incident triangles, scaled by the area mass density.

Dissipation Shells dissipate energy via flexural oscillations. To that end we complete our model with an *optional* damping force proportional to $(\dot{\theta} - \dot{\bar{\theta}})\nabla\theta$ where $\dot{\bar{\theta}} = 0$ for elastic deformations but is in general non-zero for plastoelastic deformations. This is consistent with standard derivations of Rayleigh-type damping forces using the strain rate tensor, as discussed by Baraff and Witkin[4].

Discussion Our proposed discrete flexural energy, (2), generalizes on published energies for (*flat*) plates both continuous and discrete: (a) Ge *et al.*[15] presented a geometric argument that the stored energy of a continuous inextensible plate has the form $\int_\Omega c_H H^2 + c_K K dA$ for material-specific coefficients c_H and c_K; (b) Haumann[20] used a discrete hinge energy, similarly Baraff and Witkin[4] used a discrete constraint-based energy, of the form $W_B(\mathbf{x}) = \sum_{\bar{e}} \theta_e^2$. Our approach generalizes both (a) and (b), and produces convincing simulations beyond the regime of thin plate and cloth models (see Section 4).

Our approach can also be viewed within the framework laid out by Terzopoulos *et al.*[25]: we focus on the second fundamental form, choose a computationally convenient and geometrically intuitive norm, and propose a simple, effective discretization.

Our novel formulation has three salient features: (a) the energy is invariant under rigid body transformation of both the undeformed and the deformed shape: our system conserves linear and angular momenta; (b) the piecewise nature of our geometry description is fully captured by the purely intrinsic membrane terms, and the purely extrinsic bending term; most importantly, (c) it is *simple* to implement.

3. Implementation

We encourage readers to implement this novel approach to simulating shells as follows: take working code for a thin plate or cloth simulator (*e.g.*, as presented by Baraff and Witkin[4]), and replace the bending energy with (2). From an implementation point of view, this involves minimal work. For example, consider that Baraff and Witkin[4] already implemented all the computations relating to θ_e. The key hurdle is that the undeformed configuration, $\bar{\mathbf{x}}$, is represented in $(x,y) \in \mathbb{R}^2$ coordinates, thus kinematically imposing a *flat* shape. One could augment this with explicitly-stored undeformed-angles, $\bar{\theta}_e$, but this would work only for developable surfaces. Any surface which cannot be unfolded into a flat sheet—a surface with intrinsic curvature, such as a hat or a car body—requires a more complete treatment than this. Instead, we express $\bar{\mathbf{x}}$ in coordinates $(x,y,z) \in \mathbb{R}^3$, *i.e.*, *not* restricting ourselves to planar undeformed configuration. Consequently, the undeformed configuration is in general curved, and we must duplicate the code that computes θ_e to also compute $\bar{\theta}_e$. Rereading Baraff and Witkin's paper with these changes in mind, it is immediately clear that these modification require just a few hours of work.

As part of ongoing and future research, our priorities in implementing our simulator are extendibility and ease of implementation. We have made several design choices to aid in numerical robustness and to avoid bugs in implementing our formulas:

Newmark Time Stepping We adopt the Newmark scheme[24] for ODE integration,

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

where Δt_i is the duration of the i^{th} timestep, $\dot{\mathbf{x}}_i$ and $\ddot{\mathbf{x}}_i$ are configuration velocity and acceleration at the beginning of the i^{th} timestep, respectively, and β and γ are adjustable parameters linked to the accuracy and stability of the time scheme. Newmark is either an explicit ($\beta = 0$) or implicit ($\beta > 0$) integrator: we used $\beta = 1/4$ for final production, and $\beta = 0$ to aid in debugging. Newmark gives control over numerical damping via its second parameter γ. We obtained the best results by minimizing numerical damping ($\gamma = 1/2$); this matches Baraff and Witkin's observation that numerical damping causes undesirable effects to rigid body motions. Our shell model works also with other time steppers, *e.g.*, the implicit scheme used by Baraff and Witkin[4]. We draw the reader's attention to West *et al.*[26], who demonstrate the numerical advantages of the Newmark scheme.

Automatic Differentiation The use of an explicit integrator necessitates the evaluation of energy gradients, or forces, with respect to vertex DOFs. Formulae for the gradients of edge-length and area are easily found in the literature[13]; the gradient of the dihedral angle requires more work, but can still be derived by hand. Since our goal is to ease implementation and debugging of new, experimental energies, we chose to use an automatic differentiation (AD) technique.

The use of an *implicit* integrator necessitates evaluation of *force* gradients with respect to vertex DOFs, *i.e.*, we need formulae for *second* derivatives of energy. Deriving such formulae is cumbersome and error-prone, consequently we used AD, a technique for augmenting software with derivative computations[12]. The technique is based on the observation that every computational algorithm can be written as the composition of simple, easily differentiable, steps to which the chain rule can be applied. AD is not new to graphics[16, 21]. Our AD code is available at http://multires.caltech.edu/software.

The salient features of our AD implementation are: (a) it differentiates directly with respect to vector (not scalar) unknowns; (b) it uses C++ type-checking to ensure both efficiency and completeness of differentiation. Although there are several good AD libraries publicly available[1, 3], we opted for implementing this simple set of classes specially for differentiation with respect to vector variables.

We define two classes, Scalar and Vector, representing *independent* scalar and vector values respectively. The related classes DScalar and DVector represent *dependent* quantities; these carry a tuple *(scalar value, vector-valued derivative)* and *(vector value, matrix-valued derivative)* respectively. The standard algebraic operators are overloaded to inter-operate between the classes, with a special restriction on assignment: dependent quantities may not be assigned to independent variables, and vice-versa. This condition ensures both correctness (no dependent quantity is overlooked) and efficiency (independent quantities never compute/store derivatives). More documentation is given in the publicly-available release of our small AD library.

In a production code, we believe that hand-derived formulas would display better performance. As demonstrated in the works of Baraff and Witkin[4] and Bridson *et al.*[6], it is reasonable to explicitly take the derivatives by hand. Early in our investigation we converted our code to the automatic technique, in order to facilitate future exploration, and to learn more about the technique; although we did not compare timings of the hand- and automatic-techniques, in our research code they appeared to run at comparable speeds, and we opted for the convenience of AD: the actual performance degradation was well worth the guaranteed consistency between energy, forces, and force gradients for our research purposes.

4. Results

We exercised our implementation on three problems: fixed beams, falling hats, and pinned paper (see accompanying animations on the electronic proceedings and at http://multires.caltech.edu/pubs/DS-CDROM). Computation time, on a 2GHz Pentium 4 CPU, ranged from a few minutes to a few hours. In light of the discussion in Section 3, we expect an optimized implementation of our method to be as efficient as state-of-the-art cloth simulators.

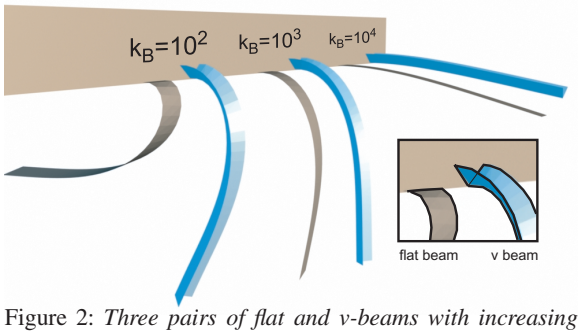


Figure 2: *Three pairs of flat and v-beams with increasing flexural stiffness (left to right) of 100, 1000, and 10000. The flexural energy coefficient has a high dynamic range; extreme values (from pure-membrane to near-rigid) remain numerically and physically well-behaved. Observe that increasing flexural stiffness augments structural rigidity. Compare the behavior of beams: the non-flat cross section of the v-beam contributes to structural rigidity, especially for low flexural stiffness.*

Beams We pinned to a wall one end of a v-beam, and released it under gravity. Figure 2, and the video, demonstrate the effect of varying flexural stiffness on oscillation amplitude and frequency. Higher flexural stiffness gives higher structural rigidity. The curved undeformed shape of a v-beam gives qualitatively and quantitatively different behavior than a flat beam. Compare: hold a simple paper strip by its end; repeat after folding a v-shaped cross-section.

Elastic hats We dropped both real and virtual hats and compared (see Figure 1): the deformation is qualitatively the same, during impact, compression, and rebound. Adjusting the damping parameter, we capture or damp away the brim's

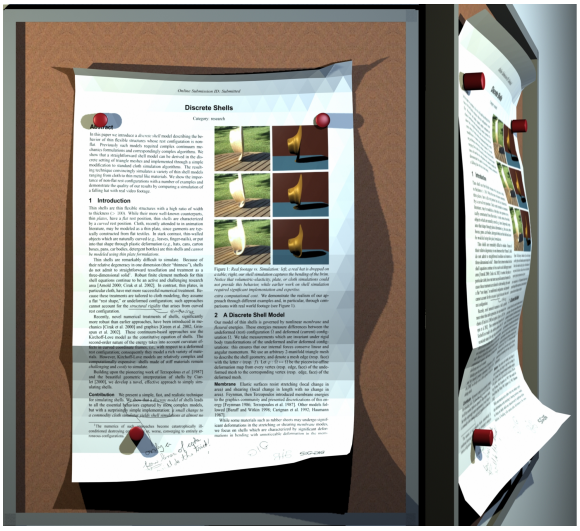


Figure 3: *Modeling a curled, creased, and pinned sheet of paper: by altering dihedral angles of the reference configuration, we effect plastic deformation. While the rendering is texture-mapped we kept flat-shaded triangles to show the underlying mesh structure (see also the color plate).*

vibrations. Adjusting the flexural stiffness, we can make a hat made of hard rubber or textile (see the videos of a nearly-rigid hat and a floppy hat).

Plastoelasticity As discussed in the early work of Kergosien *et al.*[22], a compelling simulation of paper would require a mechanical shell model. Using our simple shell model, we can easily simulate a sheet of paper that is rolled, then creased, then pinned (see Figure 3). Here the physics require *plastic* as well as elastic deformations. We begin with a flat surface, and gradually increase the undeformed angles, $\bar{\theta}_e$. Notice: modifying the *undeformed* configuration effects a *plastic* deformation. The kinematics of changing $\bar{\theta}_e$ span only physically-realizable bending, *i.e.*, *inextensible* plastic deformations. In contrast, directly modifying $\bar{\mathbf{x}}$ could introduce plastic deformations with unwanted membrane modes. We introduced elastic effects by applying three pin constraints to the *deformed* configuration. Observe the half-crease on the left side. The energy of the undeformed state is no longer zero! The (plastically-deformed) left and (untouched) right halves have incompatible undeformed shapes, consequently the undeformed configuration is *not* stress-free.

5. Conclusions

We introduced a novel discrete model of thin shells for computer animation, generalizing earlier discrete models of thin plates, while complementing contemporaneous developments in cloth simulation. Earlier treatments of shells require an understanding of advanced finite element techniques and significant implementation effort. To our knowledge, our work here is the first effort to geometrically *derive* a discrete model for thin shells aimed at computer animation. We demonstrate this model with animations of *stiff* bendable shells such as hats and paper. Notice that the independent work by Bridson *et al.*[6], in these proceedings, offers compelling results of complex wrinkles and folds in clothing using a similar formulation. Together these works reinforce each other, spanning soft and stiff shells.

From an implementation point of view, the difference with earlier work in cloth modeling seems trivial. The key is a series of significant observations, each with minimal (but important!) impact on implementation. The kinematic setup must allow for *intrinsic* curvature: many interesting surfaces are not developable. The flexural energy must measure *only extrinsic* curvature: mean curvature and *not* Gaussian curvature. Keeping membrane and bending energies separate is reasonable, and desirable. The discrete operators can (and should) be simple. Together, these give the simplest formulation of shells we have seen that produces convincing animations. Finally, we also describe our choice of a Newmark implicit integrator, with well-established symplectic properties, and our use of automatic force derivation to facilitate development and further research.

Our simple model captures the characteristic behaviors of shells, including flexural rigidity and crumpling; visually, animations compare favorably to sophisticated shell models requiring cumbersome high-order constitutive equations and finite-element techniques. We claim that implementing a thin shell simulator for graphics applications is now practical and worthwhile.

Acknowledgments This work was supported in part by NSF (DMS-0220905, DMS-0138458, DMS-0221666, DMS-0221669, CCR-0133983, EEC-9529152, ACI-0219979) the DOE (W-7405-ENG-48/B341492), nVidia, the Center for Integrated Multiscale Modeling and Simulation, Intel, Alias|Wavefront, Pixar, Microsoft, and the Packard Foundation. We thank Jerry Marsden, Tom Duchamp, and Anastasios Vayonakis for insightful discussions, and our colleagues in the Multi-Res group for their support. We are indebted to Pierre Alliez, Ilja Friedel, and Steven Schkolne for their production assistance.

References

1. ADIFOR. Argonne National Laboratory / Rice University. http://www-unix.mcs.anl.gov/autodiff/ADIFOR/, 2002.

2. Douglas Arnold. Questions on shell theory. Workshop on Elastic Shells: Modeling, Analysis, and Computation, 2000. Mathematical Sciences Research Institute, Bekeley.

3. Autodiff.org. http://www.autodiff.org, 2002.

4. David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of SIGGRAPH*, pages 43–54, 1998.

5. Robert Bridson, Ronald P. Fedkiw, and John Anderson. Robust treatment of collisions, contact, and friction for cloth animation. *ACM Trans. on Graphics*, 21(3):594–603, July 2002.

6. Robert Bridson, Sebastian Marino, and Ronald Fedkiw. Simulation of clothing with folds and wrinkles. In D. Breen

and M. Lin, editors, *Proceedings of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2003.

7. Michel Carignan, Ying Yang, Nadia Magnenat Thalmann, and Daniel Thalmann. Dressing animated synthetic actors with complex deformable clothes. In *Proceedings of SIGGRAPH*, pages 99–104, 1992.

8. Philippe Ciarlet. *Mathematical Elasticity. Vol. III*, volume 29 of *Studies in Mathematics and its Applications*. Amsterdam, 2000. Theory of shells.

9. F. Cirak, M. Ortiz, and P. Schröder. Subdivision surfaces: A new paradigm for thin-shell finite-element analysis. *Internat. J. Numer. Methods Engrg.*, 47(12):2039–2072, 2000.

10. F. Cirak, M.J. Scott, E.K. Antonsson, M. Ortiz, and P. Schröder. Integrated modeling, finite-element analysis, and engineering design for thin-shell structures using subdivision. *CAD*, 34(2):137–148, 2002.

11. David Cohen-Steiner and Jean-Marie Morvan. Restricted delaunay triangulations and normal cycle. In *Proc. 19th Annu. ACM Sympos. Comput. Geom.*, pages 237–246, 2003.

12. George Corliss, Christèle Faure, Andreas Griewank, Laurent Hascoët, and Uwe Naumann, editors. *Automatic Differentiation of Algorithms: From Simulation to Optimization*. Springer, 2001.

13. Mathieu Desbrun, Mark Meyer, and Pierre Alliez. Intrinsic parameterizations of surface meshes. In *Proceedings of Eurographics*, pages 209–218, 2002.

14. C. Feynman. Modeling the appearance of cloth. MSc thesis, MIT, 1986.

15. Z. Ge, H. P. Kruse, and J. E. Marsden. The limits of hamiltonian structures in three-dimensional elasticity, shells, and rods. *Journal of Nonlinear Science*, 6:19–57, 1996.

16. Michael Gleicher. *A Differential Approach to Graphical Manipulation (Chapter 5)*. PhD thesis, 1994.

17. Alfred Gray. *Modern Differential Geometry of Curves and Surfaces*. Second edition. CRC Press, 1998.

18. Seth Green, George Turkiyyah, and Duane Storti. Subdivision-based multilevel methods for large scale engineering simulation of thin shells. In *Proceedings of ACM Solid Modeling*, pages 265–272, 2002.

19. Eitan Grinspun, Petr Krysl, and Peter Schröder. CHARMS: a simple framework for adaptive simulation. *ACM Transactions on Graphics*, 21(3):281–290, July 2002.

20. R. Haumann. Modeling the physical behavior of flexible objects. In *Topics in Physically-based Modeling,* Eds. Barr, Barrel, Haumann, Kass, Platt, Terzopoulos, and Witkin*, SIGGRAPH Course Notes*. 1987.

21. Michael Kass. CONDOR: constraint-based dataflow. In *Proceedings of SIGGRAPH*, pages 321–330, 1992.

22. Y. L. Kergosien, H. Gotoda, and T. L. Kunii. Bending and creasing virtual paper. *IEEE Computer Graphics and Applications*, pages 40–48, 1994.

23. Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In Hans-Christian Hege and Konrad Polthier, editors, *Visualization and Mathematics III*, pages 35–57. Springer-Verlag, Heidelberg, 2003.

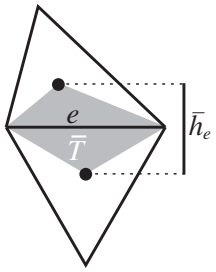
24. N. M. Newmark. A method of computation for structural dynamics. *ASCE J. of the Engineering Mechanics Division*, 85(EM 3):67–94, 1959.

25. Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Proceedings of SIGGRAPH*, pages 205–214, 1987.

26. M. West, C. Kane, J. E. Marsden, and M. Ortiz. Variational integrators, the newmark scheme, and dissipative systems. In *International Conference on Differential Equations 1999*, pages 1009 – 1011, Berlin, 2000. World Scientific.

Appendix: Derivation of the Flexural Energy

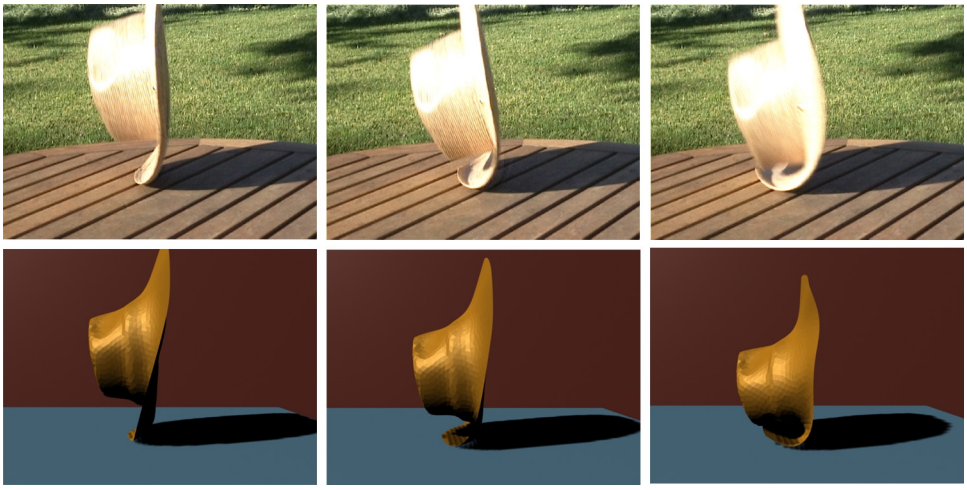
We derive the discrete, integral *mean-curvature squared* operator as follows. We first partition the undeformed surface into disjoint union of diamond-shaped tiles, $\bar{T}$, associated to each mesh edge, e, as indicated on the side figure (following Meyer *et al.*[23], one can use the barycenter of each triangle to define these regions—or alternatively, the circumcenters). Over such a diamond, the mean curvature integral is $\int_{\bar{T}} \bar{H} d\bar{A} = \bar{\theta}\|\bar{e}\|$ (see Cohen-Steiner and Morvan[11] for a proof). A similar argument leads to: $\int_{\bar{T}} (H \circ \varphi - \bar{H}) d\bar{A} = (\theta - \bar{\theta})\|\bar{e}\|$. Using the notion of area-averaged value from Meyer *et al.*[23], we deduce that $(H \circ \varphi - \bar{H})|_{\bar{T}} = (\theta - \bar{\theta})/\bar{h}_e$, where $\bar{h}_e$ is the span of the undeformed tile, which is one sixth of the sum of the heights of the two triangles sharing $\bar{e}$. For a sufficiently fine, non-degenerate tessellation approximating a smooth surface, the average over a tile (converging pointwise to its continuous counterpart) *squared* is equal to the squared average, leading to: $\int_{\bar{T}} (H \circ \varphi - \bar{H})^2 d\bar{A} = (\theta - \bar{\theta})^2 \|\bar{e}\|/\bar{h}_e$.



Simplification Interestingly, for a simplified version of the above formula of the form $(\theta - \bar{\theta})^2\|\bar{e}\|$, the energy functional becomes dependent only on its piecewise planar geometry *not* on the underlying triangulation. This is appealing in that a material's physical energy should depend on its shape, *not* on the discretization of the shape. However, there is no discretization of (1) that simultaneously is (a) dependent only on the geometry *not* its triangulation, and (b) converges to its continuous equivalent under refinement. The area integral of (1) is, in general, unbounded for a piecewise planar geometry! A discrete energy satisfying both (a) and (b) may exist for smoother surfaces, but our focus is piecewise planar (triangle mesh) geometry. In our falling hat simulations, we found that the simplified energy, which has property (a) but *not* (b), led to satisfactory results; however, our meshes were quasi-uniformly sampled and not adaptively refined. In general we advocate the use of (2), which is only slightly more complex to implement but is robust in the presence of nonuniform sampling.

Accuracy Following the argument found in Meyer *et al.*[23], there may be numerical advantages in using circumcenters instead of barycenters for the definition of the diamond tiles (except in triangles with obtuse angles). This affects the definition of $\bar{h}_e$ and of the lumped mass. Since we only need to compute these values for the undeformed shape, the implementation and performance of initialization code would be affected.

Using our novel formulation for discrete shells, we model a curled, creased, and pinned sheet of paper. By altering dihedral angles of the reference configuration, we effect plastic deformation. While the rendering is texture-mapped we use flat-shading of the triangles to emphasize the discrete structure of the underlying mesh. The final shape is fully simulated; the artist indicates the curl radius, the crease sharpness, and the pin positions.



Real footage vs. Simulation: top, a real hat is dropped on a table; bottom, our shell simulation captures the bending of the brim. Notice that volumetric-elasticity, plate, or cloth simulations could not capture this behavior, while earlier work on shell simulation required significant implementation and expertise. Movie clips are included in the electronic proceedings and at http://multires.caltech.edu/pubs/DS-CDROM.