Summary of Discrete Shells

November 3, 2024

1 Introduction

Thin shells are flexible structures with high width-to-thickness ratios, characterized by curved undeformed configurations. Traditional models for simulating thin shells require complex continuum mechanics, but this paper introduces a simplified discrete shell model based on triangle meshes, offering efficient simulations with realistic behavior for materials from paper to metal.

2 Discrete Shell Model

This model of thin shells uses nonlinear *membrane* and *flexural* energies, which measure differences between the undeformed configuration $\overline{\Omega}$ and deformed configuration Ω. The deformation is described by φ which maps faces of the undeformed mesh to faces of the deformed mesh.

The total energy is given by:

$$W = W_M + k_B W_B \tag{1}$$

where k_B is the bending stiffness.

2.1 Membrane Energy

Resists stretching and shearing by considering local changes in area and length. These are intrinsic deformations (local changes in area and/or length).

Thin shells have mostly *isometric* deformations, i.e., possibly significant deformations in bending but unnoticeable deformation in the membrane modes. Most membrane models for triangle meshes satisfy the small membrane-strain assumption if you choose a suitably large membrane stiffness coefficient.

2.2 Flexural Energy

Resists bending using differences in the mean curvature between deformed and undeformed configurations, ensuring invariance under rigid-body transformations.

Shell models use the difference of the second fundamental forms (which inform on curvature) in the deformed and undeformed configurations, pulling back the deformed configuration onto the undeformed configuration.

2.2.1 Discrete Differential Geometry Recap

- Shape operator S:
  - Derivative of the Gauss map and measures local curvature at a point on a smooth surface by measuring the change in surface normal.
- $\mathrm{Tr}(S) : \kappa_1 + \kappa_2$
  - Sum of eigenvalues of the shape operator S.
- Principal curvatures κ_1, κ_2: Directions with the most change in surface normal.
- Mean curvature H: average curvature at a surface point, integrated over all directions.
  - $$H = \frac{\kappa_1 + \kappa_2}{2} = \frac{\mathrm{Tr}(S)}{2} \tag{2}$$

2.2.2 Bending Energy

Our bending energy is an extrinsic measure of the difference between the shape operator evaluated on the deformed and undeformed surface. We can now express this difference as the *squared difference of mean curvature:*

$$[\mathrm{Tr}(\varphi^* S) - \mathrm{Tr}(\overline{S})]^2 = 4(H \circ \varphi - \overline{H})^2 \tag{3}$$

As previously described, φ maps the undeformed mesh to the deformed mesh. φ^* pulls the shape operator onto the undeformed shape $\overline{\Omega}$. It does **not** change the shape, but allows us to compare the two shapes by bringing them to the same reference frame by telling us which point on the *deformed* surface corresponds to which point on the *undeformed* surface.

To find the total (continuous) flexural energy, we need to integrate over the whole domain.

$$\int_{\overline{\Omega}} 4(H \circ \varphi - \overline{H})^2 d\overline{A} \tag{4}$$

Discretizing it because we're using a discrete mesh:

$$W_B(\mathbf{x}) = \sum_e (\theta_e - \overline{\theta}_e)^2 \frac{\|\overline{e}\|}{\overline{h}_e} \tag{5}$$

- θ_e and $\overline{\theta}_e$: the dihedral angle of edge e in the deformed and undeformed configurations, respectively.
- $\overline{h}_e = \frac{1}{3}\left(\frac{h_1+h_2}{2}\right)$, where h_1 and h_2 are the heights of the two triangles adjacent to edge e.

2.3 Dynamics

The dynamics of the thin shell model are governed by the following principles and methods:

- The model's motion is determined by the ordinary differential equation:

  $$\ddot{x} = -M^{-1}\nabla W(x) \tag{6}$$

  where x represents the **vertices of the deformed shell configuration**, M is the **mass matrix**, and $W(x)$ is the **total energy**.

- **Mass Distribution**: For computational simplicity, the mass is assumed to be "lumped" at each vertex, resulting in a diagonal mass matrix M.

  - The **area mass density** ρ represents the mass per unit area of the material. If a surface element has an area A and a total mass m, then ρ is defined as:

  $$\rho = \frac{m}{A} \tag{7}$$

  In the context of the shell model, ρ is used to calculate the mass assigned to each vertex by scaling the area of triangles incident to that vertex.

  If A_i represents the area of the i-th triangle incident to a given vertex, then the mass m assigned to that vertex is:

  $$m = \frac{\rho}{3}\sum_i A_i \tag{8}$$

  where the summation is over all triangles i that are incident to the vertex. This formulation divides the area contribution equally among the three vertices of each triangle.

- **Dissipation**: Thin shells lose energy through oscillations, particularly in flexural modes. To model this, a damping force is applied, proportional to

  $$(\dot{\theta} - \dot{\bar{\theta}})\nabla\theta \tag{9}$$

  where $\dot{\bar{\theta}} = 0$ for purely elastic deformations. **For plastoelastic deformations, $\dot{\bar{\theta}}$ can be non-zero.**

  This approach to damping aligns with Rayleigh-type damping methods and incorporates the strain rate tensor, as discussed in prior work by Baraff and Witkin.

3 Implementation

A practical implementation can be achieved by modifying existing cloth simulators. The paper suggests taking working code for a cloth simulator and replacing the bending energy with W_B as described in 2.2.2.

3.0.1 Undeformed Angles

The key change is that in the case of the thin plate model, the undeformed configuration $\overline{x}$ is represented in $(x, y) \in \mathbb{R}^2$, imposing a flat shape.

The paper duplicates the code that computes θ_e to also compute $\overline{\theta_e}$ to generalize the model to surfaces with intrinsic curvature. However, since we are working with a developable surface (a surface that can be unfolded into a flat sheet), we can just explicitly store undeformed angles θ_e and save ourselves from more trouble.

3.0.2 Time Stepping

The paper adopts the **Newmark integration scheme** for solving the equations of motion in shell dynamics.

- **Newmark Scheme**: The position x_{i+1} and velocity $\dot{x}_{i+1}$ at the next time step:

$$x_{i+1} = x_i + \Delta t\, \dot{x}_i + \Delta t^2 \left(\frac{1}{2} - \beta\right) \ddot{x}_i + \beta \Delta t^2 \ddot{x}_{i+1} \tag{10}$$

$$\dot{x}_{i+1} = \dot{x}_i + \Delta t\, ((1-\gamma)\ddot{x}_i + \gamma \ddot{x}_{i+1}) \tag{11}$$

  where Δt is the time step, and β and γ are adjustable parameters that control the stability and damping properties of the scheme.

- **Parameter Choices**: For the final simulations, the authors set $\beta = \frac{1}{4}$ (implicit formulation) and $\gamma = \frac{1}{2}$ to minimize numerical damping, consistent with Baraff and Witkin's recommendations for reducing unwanted damping in rigid-body motion. During debugging, they set $\beta = 0$ to use an explicit formulation.

- **Numerical Damping**: Choosing $\gamma = \frac{1}{2}$ minimizes numerical damping, essential for realistic rigid-body motion. Lower damping ensures that shells maintain natural flexural oscillations, leading to more physically accurate animations.

Baraff and Witkin's time integration should also work, as argued by the paper, which is **implicit backward Euler**.

$$\begin{aligned} x_{i+1} &= x_i + \Delta t\, \dot{x}_{i+1} \\ \dot{x}_{i+1} &= \dot{x}_i + \Delta t\, \ddot{x}_{i+1} \\ M\, \ddot{x}_{i+1} &= F(x_{i+1}) - D\, \dot{x}_{i+1} \end{aligned} \tag{12}$$

Note: This implicit method stabilizes stiff forces, allowing larger time steps.

3.0.3 Automatic Differentiation

We now have to take second derivatives to solve the ODEs since the integration scheme is implicit.

Baraff and Witkin took these Hessians by hand, but this is a bit of an issue since they depend on dihedral angles. So we don't want to do that.

Luckily the authors of the paper published their small AD library that they used for the paper here. We might be allowed to use it.