Second-Order Finite Elements for Deformable Surfaces (Supplementary Material)

QIQIN LE, Shanghai Qi Zhi Institute, China
YITONG DENG, Dartmouth College, United States of America
JIAMU BU, Tsinghua University, China
BO ZHU, Georgia Institute of Technology, United States of America & Dartmouth College, United States of America
TAO DU, Tsinghua University, China & Shanghai Qi Zhi Institute, China

ACM Reference Format:
Qiqin Le, Yitong Deng, Jiamu Bu, Bo Zhu, and Tao Du. 2023. Second-Order Finite Elements for Deformable Surfaces (Supplementary Material). In *SIGGRAPH Asia 2023 Conference Papers (SA Conference Papers '23), December 12–15, 2023, Sydney, NSW, Australia.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3610548.3618186

1 FORMULA OF BENDING ENERGY AND ITS DERIVATIVES

We provide detailed formula of our bending energy defined in the paper. First we define point-wise mean discrete curvature on a piecewise smooth curved triangular surface:

$$H_d(u,v) = H_a(u,v) + \frac{1}{|T|}\int_e \theta ds, \tag{1}$$

where H_a is the analytical mean curvature of a curved element at (u,v). T and e are the blue diamond domain and the middle edge in Fig. 1.θ is the local bending angle on e. Omitting the bending stiffness, we define our bending energy on T as

$$E_b = \int_T H_d^2(u,v)dudv. \tag{2}$$

Because θ goes to zero as the discretization gets refined, we omit the term including $H_d\int_e \theta ds$. But we keep the term with $(\int_e \theta ds)^2$ in order to separately handle face bending and edge bending. Also we approximate $(\int_e \theta ds)^2$ with $(\int_e \theta^2 ds)$. Therefore, we get an approximation that could be easily implemented.

$$E_b \approx \int_T H_a^2 dudv + \frac{1}{|T|}\int_e \theta^2 ds. \tag{3}$$

We use Gaussian quadrature to approximate the integral in (3).

$$(3) \approx E_b^* = \sum_i w_i H_a^2(x_i) + \frac{1}{|T|}\sum_j w_j \theta^2(x_j), \tag{4}$$

where x_i, x_j and w_i, w_j correspond to quadrature points and weights on T and e. In our implementation, we applied mid-edge-point-rule of triangles for quadratic estimation on T. For edges, we took two points for quadratic estimation. Its gradient would simply be

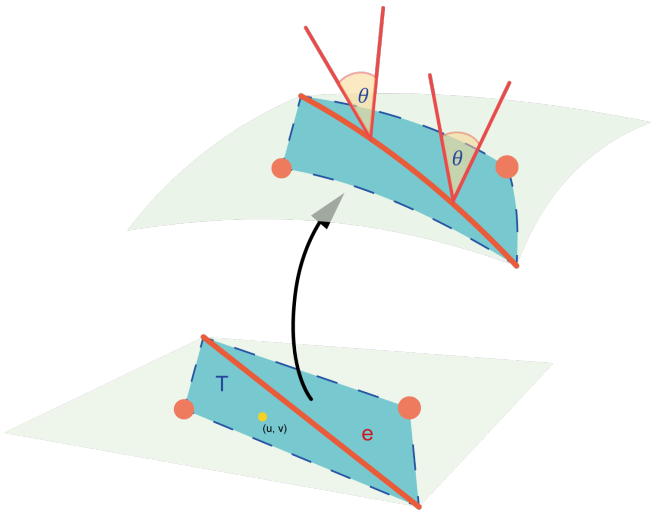

Fig. 1. The diagram of the bending energy definition

$$\nabla E_b^* = \sum_i w_i \nabla H_a^2(x_i) + \frac{1}{|T|}\sum_j w_j \nabla\theta^2(x_j). \tag{5}$$

We could use the same way to compute its Hessian, but in practice it suffices to estimate the Hessian by its gradient.

$$HessE_b^* \approx \sum_i 2w_i \nabla H_a(x_i)\nabla H_a^T(x_i) + \frac{1}{|T|}\sum_j 2w_j \nabla\theta(x_j)\nabla H_a^T(x_i), \tag{6}$$

The scheme ensures the positive-definiteness of the estimated matrix. We found enough to solve Newtonian iterations.

2 DEGREE-OF-FREEDOM LOSS IN LINEAR ELEMENTS

We point out a subtle but significant difference between first-order and second-order finite elements in this virtual node algorithm. In the virtual node scheme, each l_i that splits T_i introduces both virtual nodes and constraints, and we can view their number difference as the degrees of freedom reserved for expressing the curve motion. By inducting on $\{l_i\}$, we conclude that this virtual node algorithm with first-order elements assigns only three degrees of freedom for the whole curve, regardless of how many T_i are involved. A similar

SA Conference Papers '23, December 12–15, 2023, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0315-7/23/12.
https://doi.org/10.1145/3610548.3618186

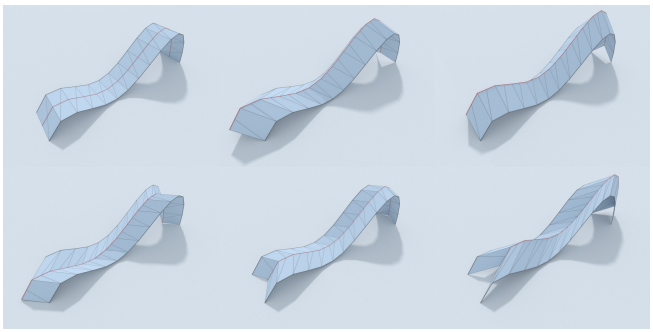

Fig. 2. A paper strip with a crease (red curve) parallel to its long boundary simulated using first-order elements. Dirichlet boundary conditions fix nodes on the two long boundaries and prescribe the kinematic motion of one end of the crease. All other nodes along the crease are supposedly free, independent nodes.

derivation shows that the curve consists of $O(|\{l_i\}|)$ degrees of freedom when second-order elements are used.

The difference seems striking at first glance. In particular, it implies that the curve will be much stiffer when first-order elements are used, deteriorating the previously mentioned locking issue. The inherent reason behind this phenomenon is that first-order elements impose an affine transform that locks the deformation gradient for every point from the same polygon. Fig. 2 uses a one-layer strip to present a figurative explanation: when the curve is parallel to the long boundary of the strip in the rest shape, it is destined to remain parallel under affine transforms. Therefore, the only degrees of freedom left for this curve are trivial translations of the deformed boundary. This observation gives us another motivation for advocating higher-order elements in thin-shell simulation.

3 MORE EVALUATIONS

Anisotropic triangulation. Anisotropic triangulation generates highly skewed triangles that hurt the performance of the finite element method, as its estimation error is bounded by the diameter of each finite element and the polynomial degrees of each basis [Johnson 2012]. The situation deteriorates when coupled with first-order triangular elements because of their piecewise-constant deformation gradients. We hypothesize that using second-order elements can remedy this problem. To verify this, we consider a model problem of a synthetic surface and discretize its (u, v) parameter domain Ω into 4×16 and 16×4 uniform rectangles, respectively, and each rectangle is further divided into two triangles using its diagonal. This way, we obtained two anisotropic meshing with orthogonal orientations. We then used first-order and second-order elements to reconstruct the deformation gradient field on both triangulations and visualized their residual errors in Fig. 3. For first-order elements, the anisotropic orientation affects the reconstruction results significantly. In particular, we observed much larger errors when assigning only 4 triangles along the u direction (Fig. 3 bottom right), as z is a cubic function of u and therefore difficult to approximate with piecewise linear functions. On the other hand, the influence of anisotropic meshing is much milder for second-order

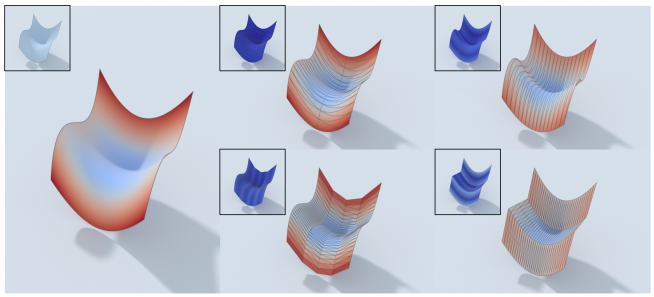

Fig. 3. Left: a synthetic surface (small inset) $\mathbf{x}(u, v)$ defined as $x = u$, $y = v$, $z = 5.4u^3 + 1.8v^2 - 0.6u$ on a squared domain $\Omega = [-0.5, 0.5]^2$. The color on the surface represents the magnitude of its Jacobian w.r.t. (u, v) normalized into $[0, 1]$ with blue and read indicating 0 and 1, respectively. Right: We discretize the surface with second-order (top row) and first-order elements on two anisotropic resolutions (middle and right columns). The red-blue color indicates the normalized magnitude of the Jacobian constructed from the finite elements, with its absolute error plotted in the small insets.

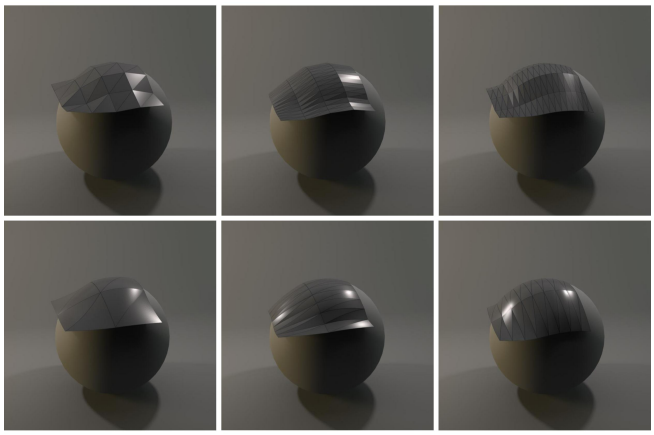

Fig. 4. Cloth falling on a sphere. Top row are obtained by first-order elements with respective resolutions of 4×4, 4×16, and 16×4 (left to right). Bottom row are obtained by second-order elements with respective resolutions of 2×2, 2×8, and 8×2, matching the number of DoFs of its counterpart.

elements, as can be seen from their similar residual errors from two triangulations (Fig. 3 top row).

Next, we evaluate the effects of anisotropic meshes in a dynamic setting. We again simulated the previously described sphere example but with anisotropic discretization and presented the results in Fig. 4. Simulation with first-order elements exhibits mesh-dependent artifacts of obvious piecewise-linear edges (Fig. 4 top row). On the other hand, simulation with second-order elements produces visually similar results regardless of anisotropic orientations (Fig. 4 bottom row). To summarize, the experiments have confirmed our speculation that, when simulating deformable codimension-one surfaces, second-order elements are more tolerant of anisotropicity in triangulation than first-order elements. We attribute this property to the expressiveness of quadratic basis functions and their spatially varying deformation gradients within each element.

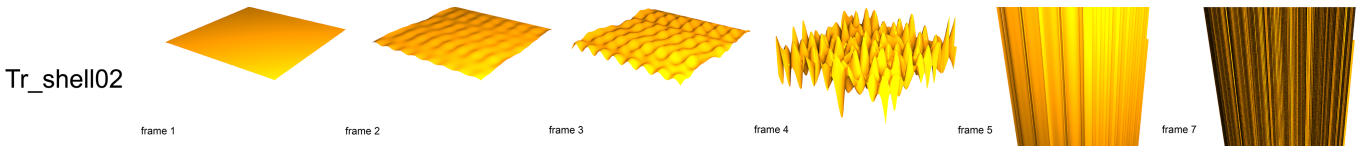

Fig. 5. *Comparisons with* `tr_shell02` This is the extension of Fig.8 in the paper. `tr_shell02` failed to solve after 7 frames.

REFERENCES

Claes Johnson. 2012. *Numerical Solution of Partial Differential Equations by the Finite Element Method.* Courier Corporation.