Adaptive Cloth Simulation

Robert Wang

Advised by Jessica Hodgins

School of Computer Science
Carnegie Mellon University
`rywang@andrew.cmu.edu`

May 2nd, 2002

Abstract

Cloth has traditionally been simulated on a static mesh composed of a uniform grid of nodes. Since a coarse mesh cannot generally capture fine wrinkles, realistic cloth simulation relies on a fine grid of nodes that is expensive to simulate. However, fine wrinkles are often localized to relatively small portions of a mesh. We introduce an adaptive method that locally refines cloth where wrinkles tend to appear and simplifies regions where wrinkles have disappeared.

Recent work has produced adaptive methods for simple models of cloth and related simulations. This project extends to more sophisticated models of cloth. In particular, this work introduces an adaptive model supporting implicit integration.

1 Introduction

Digital characters in computer animation are often draped in sheets of polygonal clothing. Cloth simulation refers to physically based modeling of cloth for realistic animation.

Cloth has been addressed both as a problem in deformable surfaces, drawing upon finite element methods [11], and as a simple particle system modeled with masses and springs [10]. Cloth may be dynamically animated or statically draped (via energy minimization) [2]. Recent work has chosen to model cloth as a mass-spring system where a grid of nodes (particles) is connected by spring-like forces defined on edges or faces. The nodes represent lumped masses while the springs define how the cloth stretches, shears, and bends.

At its heart, cloth simulation is a time-varying partial differential equation which after discretization, is numerically solved as an ordinary differential equation [1]:

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1}(-\frac{\partial E}{\partial x} + \mathbf{F})$$

where $\mathbf{x}$ represents the position of nodes with masses in diagonal matrix $\mathbf{M}$, internal energy described by E and external forces described in $\mathbf{F}$. Cloth is characterized by its resistance to stretch and permissiveness to bending, resulting in a “stiff” numerical system that quickly becomes unstable for large time steps. Recent research has concentrated on how to numerically stabilize cloth simulation. Modern cloth models [1, 3] rely on an implicit integration approach to time step the simulation.

One of the limitations of current cloth models [1, 3] is that cloth may only bend at the pre-defined edges of a static mesh. To simulate the detailed bending patterns of fine wrinkles, a fine grid of nodes must be used. However, cloth does not wrinkle uniformly. Wrinkles often are localized to one part of cloth at a time. An adaptive approach attempts to guess where new wrinkles may form (so as to introduce a finer grid of nodes where needed) and simplifies areas where wrinkles are not present.

The challenge of an adaptive model for cloth simulation is to maintain consistent behavior across nodes and forces defined at different levels of detail. Energy, mass, and momentum must be conserved during refinement. Problems often arise at junctions where elements from different levels meet (t-junctions). At these points of partial refinement, both physical and visual discontinuities may arise.

1.1 Organization

The next section describes related work in adaptive cloth simulation. Section 3 outlines the proposed method and describes our refinement criteria. In section 4 we briefly review the cloth model that we adopted. Section 5 justifies our basic method of multi-level force computation while section 6 extends the discussion to computing jacobians for implicit integration. We conclude in section 7 with some results.

2 Related Work

Hutchinson et al. [6] were the first to propose an adaptive model for cloth simulation. Hutchinson’s refinement scheme splits quad elements whenever two adjacent elements crease (and become non-planar). The model attempts to preserve the physical characteristics of cloth by doubling spring constants at each succesive level. Hutchinson’s model employs Provot’s [10] simple cloth model—more sophisticated models did not exist at the time. Hutchinson constrains t-vertices to be linearly interpolated from their parents’ positions. No explanation is given as to what is done with the forces applied to these vertices. Mesh simplification is listed as an item for future work.

Zhang and Yuen [4] proposed a global refinement scheme for simulating statically draped cloth. The cloth mesh is globally refined via triangle quadrisection whenever the simulation nears equalibrium. The approach is neither adaptive nor applicable to dynamic animation.

Recently, Villard et al.[12] proposed an adaptive model similar to Hutchinson’s, using a rectangular grid and quad-subdivision. Like Hutchinson, Villard also builds upon Provot’s [10] work and extends it with a new bend force tailored to fit his refinement scheme. Villard also doubles spring constants per level and propagates forces applied to t-vertices to their parents. No justification is provided behind the force propagation scheme. Furthermore, it is unclear how well Villard's method would extend to more elaborate cloth models [1, 3]. Villard's proposed future work includes mesh simplification, collision detection and implicit integration.

Volkov et al. [13] introduced an adaptive model extending Baraff and Witkin's cloth [1]. The technique employs the $\sqrt{3}$ (triangle) subdivision scheme [7]. An advantage of the $\sqrt{3}$ scheme is that it bypasses the problem of dealing with unnatural junctions betweens elements at different levels. However, energy conservation is not addressed in Volkov's work. It is not clear whether cloth behavior is preserved in the adaptive model.

Grinspun et al. [5] addressed adaptive simulation for finite element methods by espousing refinement of basis functions rather than of elements. However, it is not obvious how the approach could extend for particle system in which basis functions are not present.

We propose a new model for adaptive refinement of cloth that focusses on producing consistent behavior across levels by explicitly addressing conservation of energy, momentum and mass. We extend Baraff and Witkin's cloth model with a triangle quadrisection scheme. Our policy for handling forces at t-vertices is derived naturally from Baraff and Witkin's original definitions of force and energy. The requisite terms for implicit integration are derived similarly. The derivations are tailored to be easily implemented on top of a non-adaptive cloth model. Our results show side-by-side comparisons between equivalent adaptive and non-adaptive simulations.

3 Refinement Framework

In our model, forces are computed over both edges and triangles. An *active* element is an edge or a triangle on which forces are currently computed over. Refinement activates finer level elements and deactivates coarser level elements while simplification does the opposite.

3.1 Subdivision

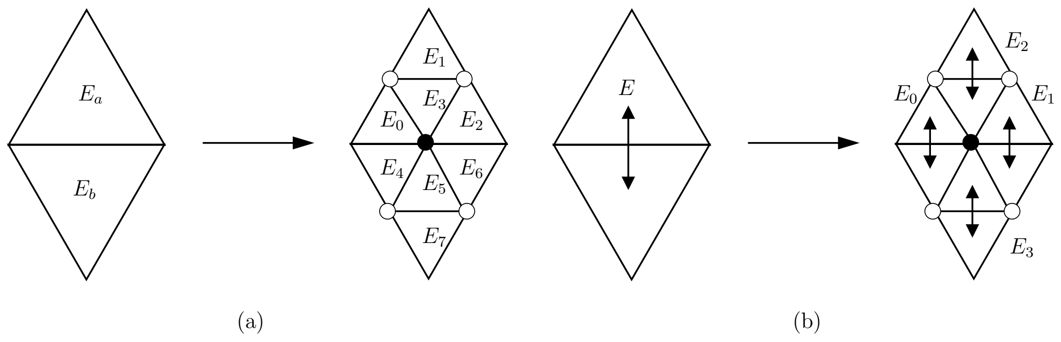

Figure 1: The effect of an edge split on energy terms defined over triangles (a) and edges (b).

Our basic refinement operator is the edge split. An edge split always activates exactly four formerly inactive child edges and deactivates the active parent edge. An edge split also requires that the child triangles of the two triangles sharing the edge to be active. Since each triangle may be involved in as many as three edge splits, an edge split does not always activate an inactive triangle. We track the number of edges requiring the child triangle to be active by reference counting splits and joins. When no edges require a triangle to be active, the triangle is deactivated and its parent is made active.

```
split(edge $e$)
    mark $e$ as inactive
    for each triangle $t_i \in \{t_1, t_2\}$ that share $e$
        for each child triangle $t$ of $t_i$
            increment active count of $t$
        decrement active count of $t_i$
    for each child edge $e_i$ of $e$
        mark $e_i$ as active
```

The join operator is the exact opposite of the edge split:

```
join(edge $e$)
    mark $e$ as active
    for each triangle $t_i \in \{t_1, t_2\}$ that share $e$
        for each child triangle $t$ of $t_i$
            decrement active count of $t$
        increment active count of $t_i$
    for each child edge $e_i$ of $e$
        mark $e_i$ as inactive
```

3.2 T-junctions

T-junctions are the places where two elements of different levels connect. A t-vertex is a finer level vertex constrained to lie on a coarser level edge so as to avoid physical and visual discontinuities. We detect and flag t-vertices during refinement and simplification.

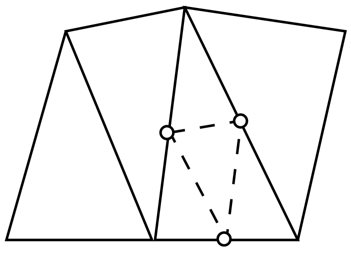

Figure 2: T-junctions and t-vertices formed by splitting an element.

Since t-vertices cannot move, something must be done about the forces applied to them. One method would be to use Lagrange multipliers to constrain t-vertices. However, Lagrange multipliers do not enforce constraints exactly and contribute to extra numerical stiffness. We chose instead to re-derive relevant forces and jacobians treating t-vertices as dependent variables (a linear combination of its two parents). This derivation yields a policy of propagating the forces to parent vertices (see Section 5).

We deal with visual discontinuities of T-vertices by rendering a smooth subdivision mesh [8] using our simulated nodes as control points.

3.3 Refinement Criteria

Our refinement criteria selects areas in a mesh that are undergoing bend. Since curvature is a pre-requisite to wrinkle formation, our refinement criteria naturally introduces detail where wrinkles are likely to form. Curvature at an edge is approximated by the angle between the normals of the two triangles sharing the edge. If the angle is above a certain threshold, we split the edge (see Figure 3).

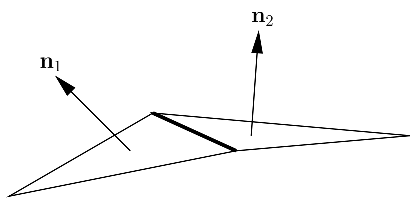

Figure 3: Refinement criteria based on the dot product of the normal across an edge

```
if $(\mathbf{n}_1 \cdot \mathbf{n}_2) < c_{split}$ then
    perform a split.
```

Simplification (unrefinement) attempts to coalesce flat portions of the mesh. For each inactive parent edge, if the maximum curvature defined over each of the child edges drops below a certain threshold, we join the triangles (see Figure 4).

```
for each edge $e$ shared by $t_i, t_j$
    if $\mathsf{max}(\mathbf{n}_i \cdot \mathbf{n}_j) > c_{join}$ then
        return
perform a join
```

A more sophisticated simplification criteria that takes in account of stretch and shear of child triangles may be employed instead. However, we found our simpler criteria to be sufficient. The simplification operator is undetectable in our examples.

4 Cloth Model

We adopted Baraff and Witkin's [1] cloth model for our simulation. Energy functions are defined through minimization of soft constraints,

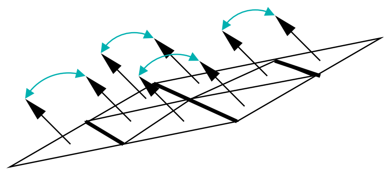

Figure 4: For each edge (darkened lines), we take the the dot product of the pair of normals across the edge.

$$E_{\mathbf{C}}(\mathbf{x}) = \frac{k}{2}\mathbf{C}(\mathbf{x})^T\mathbf{C}(\mathbf{x})$$

4.1 Conservation of Energy

Below, we describe each of these soft constraints and their required adjustments so as to conserve energy upon refinement.

- Stretching behavior is imposed by the constraint,

$$\mathbf{C}(\mathbf{x}) = a\begin{pmatrix}\|\mathbf{w}_u(\mathbf{x})\| \\ \|\mathbf{w}_v(\mathbf{x})\|\end{pmatrix}$$

  where $\mathbf{w}_u$ and $\mathbf{w}_v$ measure stress along the u and v directions. When a triangle is linearly quadrisected, the sum of the stretch energy computed over each of the child triangles is 1/4 the stretch energy of the original triangle. We compensate for this by multiplying the stretch term by 4^{level}.

- The shear constraint is defined similarly,

$$\mathbf{C}(\mathbf{x}) = a\mathbf{w_u}(\mathbf{x})^T\mathbf{w_v}(\mathbf{x})$$

  Like the stretch term, shear also decreases by a factor of 1/4 after linear quadrisection and we compensate by multiplying the shear term by 4^{level}.

- Unlike stretch and shear, the bend constraint is defined across an edge; over both of the triangles that share the edge,

$$\begin{aligned}\mathbf{C}(\mathbf{x}) &= \theta \text{ where}\\ \theta &= \cos^{-1}(\mathbf{n_1}\cdot\mathbf{n_2}) \text{ or}\\ \theta &= \sin^{-1}((\mathbf{n_1}\times\mathbf{n_2})\cdot\mathbf{e})\\ &\ \mathbf{n_1},\mathbf{n_2} \text{ are normals of triangles sharing the edge}\\ &\ \mathbf{e} \text{ is the edge as a normalized vector.}\end{aligned}$$

Under linear quadrisection, the sum of the child bend energies is twice the strength of the original. Multiplying by 2^{-level} compensates.

Given the above energy definitions, the force on particle i is defined as,

$$\mathbf{f}_i = -\frac{\partial E}{\partial \mathbf{x}_i}$$

where E is total internal energy.

The forces ($\mathbf{f}_0$) and their jacobians ($\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{f}}{\partial \mathbf{v}}$) are assembled in vector and sparse matrix form in the backwards euler step equation,

$$\left(\mathbf{I} - h\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{v}} - h^2\mathbf{M}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)\Delta\mathbf{v} = h\mathbf{M}^{-1}\left(\mathbf{f}_0 + h\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{v}_0\right)$$

where, $\mathbf{M}$ is the mass matrix; $\mathbf{v}_0$ is the velocity at the beginning of the time step; and $\Delta\mathbf{v}$, $\Delta\mathbf{x} = h(\mathbf{v}_0 + \Delta\mathbf{v})$ are the adjustments for the next time step to velocity and position respectively. The linear system above is generated and solved by the method of conjugate gradients. Collisions are handled by introducing constraints enforced by the solver [1].

4.2 Conservation of Mass and Momentum

Our lumped mass approach refines in a roughly even way by maintaining the invariant that each non-t vertex has a mass proportional to the area of the triangles adjacent to the vertex. It does not make sense for t-vertices to have mass, since their forces are propagated to their parents (see section 5). A more sophisticated approach involving Voronoi cells [9] could have been employed, but our method seemed to work fine.

Each level-0 node begins with a lumped mass of 1/2 unit per adjacent (level-0) triangle. Mass is transferred from the parents of a child vertex when the child becomes a non-t vertex. For each successive level of subdivision, $\frac{3}{2*4^{level}}$ units of mass are transferred from each parent to the child (see Figure 5).

For instance, a level-0 regular vertex begins with 3 units of mass and ends with 3/4 units upon refinement to level-1. A level-1 vertex begins with 3/4 units and ends with 3/16 units. In the limit case, when all vertices are regular, each vertex retains a quarter of its original mass on refinement. This is consistent with the fact that the triangles each vertex shares is quadrisected.

Momentum is conserved for new nodes by taking a weighted average of the velocity of the two parents. The weights should be the same as those used for mass transfer. In our case, since we're inheriting equal amounts of mass from each parent, the weights are just 1/2.

5 Force Propagation

In the original cloth model, positions of any two vertices are independent and the formula for force at a node is static for each triangle. However in our adaptive model, t-vertices

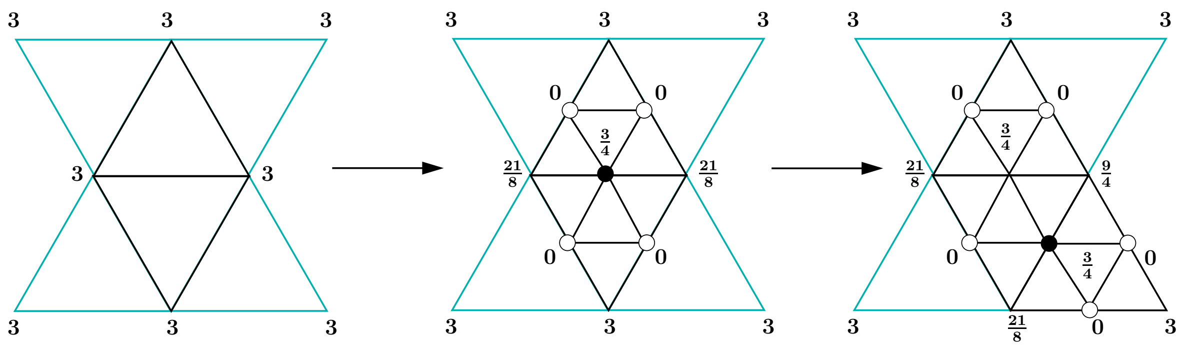

Figure 5: In this case, each edge split introduces one new non-t vertex. The second non-t vertex is produced by transitioning from a former t vertex. In both cases 3/8 units of mass are drawn from each parent. T-vertices are shown to have zero mass.

are not simulated and have no degrees of freedom–they are dependent variables.

Consider the case in Figure 6 where vertices 4, 5 and 6 are t-vertices. We derive the case for an energy constraint defined on a triangle (stretch or shear) below.

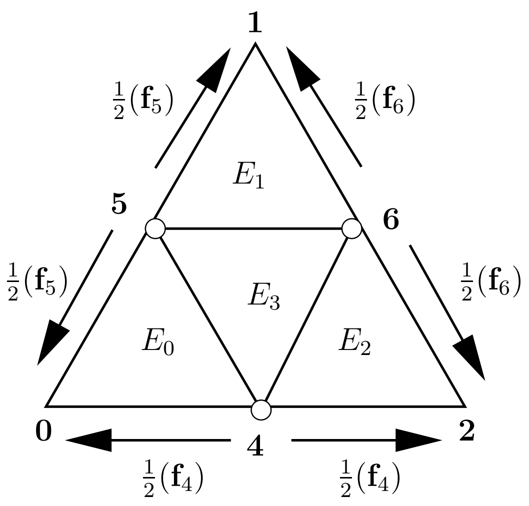
1
$\frac{1}{2}(\mathbf{f}_5)$
$\frac{1}{2}(\mathbf{f}_6)$
E_1
5
6
E_3
E_0
E_2
0
4
2
$\frac{1}{2}(\mathbf{f}_4)$

Figure 6: According to our policy, forces exerted on t-vertex i are halved and propagated to i's parents.

$$\mathbf{x}_4 = \mathbf{x}_0 + \mathbf{x}_2$$
$$\mathbf{x}_5 = \mathbf{x}_0 + \mathbf{x}_1$$
$$-\frac{\partial \mathbf{x}_4}{\partial \mathbf{x}_0} = -\frac{\partial \mathbf{x}_5}{\partial \mathbf{x}_0} = \frac{1}{2}$$

The force on a parent now involves the energy terms dependent on the parent (E_0) and the parent's two t-vertex children (E_1, E_2, E_3). Our force expression is now dynamic and dependent on both the number and the permutation of t-vertices for the given parent. Below, we show that we may recover the force on a parent vertex, taking in account of t-vertex dependencies ($\mathbf{f}_0$), from the forces on the parent and t-vertices, not taking in account of dependencies ($(\mathbf{f}_0)_{\mathbf{x}_4,\mathbf{x}_5}$, $(\mathbf{f}_4)_{\mathbf{x}_0,\mathbf{x}_5}$, $(\mathbf{f}_5)_{\mathbf{x}_0,\mathbf{x}_4}$). This derivation actually results in an intuitive policy: the forces on a t-vertex (evaluated without accounting for dependence) should be split between the two parents.

$$\begin{aligned}
\mathbf{f}_0 &= -\frac{\partial E_0}{\partial \mathbf{x}_0} - \frac{\partial E_1}{\partial \mathbf{x}_0} - \frac{\partial E_2}{\partial \mathbf{x}_0} - \frac{\partial E_3}{\partial \mathbf{x}_0} \\
(\mathbf{f}_4)_{\mathbf{x}_0,\mathbf{x}_5} &= -\left(\frac{\partial E_0}{\partial \mathbf{x}_4}\right)_{\mathbf{x}_0,\mathbf{x}_5} - \left(\frac{\partial E_2}{\partial \mathbf{x}_4}\right)_{\mathbf{x}_0,\mathbf{x}_5} - \left(\frac{\partial E_3}{\partial \mathbf{x}_4}\right)_{\mathbf{x}_0,\mathbf{x}_5} \\
(\mathbf{f}_5)_{\mathbf{x}_0,\mathbf{x}_4} &= -\left(\frac{\partial E_0}{\partial \mathbf{x}_5}\right)_{\mathbf{x}_0,\mathbf{x}_4} - \left(\frac{\partial E_1}{\partial \mathbf{x}_5}\right)_{\mathbf{x}_0,\mathbf{x}_4} - \left(\frac{\partial E_3}{\partial \mathbf{x}_5}\right)_{\mathbf{x}_0,\mathbf{x}_4}
\end{aligned}$$

where,

$$\begin{aligned}
-\frac{\partial E_0}{\partial \mathbf{x}_0} &= -\left(\frac{\partial E_0}{\partial \mathbf{x}_0}\right)_{\mathbf{x}_4,\mathbf{x}_5} - \left(\frac{\partial E_0}{\partial \mathbf{x}_4}\right)_{\mathbf{x}_0,\mathbf{x}_5} \frac{\partial \mathbf{x}_4}{\partial \mathbf{x}_0} - \left(\frac{\partial E_0}{\partial \mathbf{x}_5}\right)_{\mathbf{x}_0,\mathbf{x}_4} \frac{\partial \mathbf{x}_5}{\partial \mathbf{x}_0} \\
-\frac{\partial E_1}{\partial \mathbf{x}_0} &= -\left(\frac{\partial E_1}{\partial \mathbf{x}_5}\right)_{\mathbf{x}_0,\mathbf{x}_4} \frac{\partial \mathbf{x}_5}{\partial \mathbf{x}_0} \\
-\frac{\partial E_2}{\partial \mathbf{x}_0} &= -\left(\frac{\partial E_2}{\partial \mathbf{x}_4}\right)_{\mathbf{x}_0,\mathbf{x}_5} \frac{\partial \mathbf{x}_4}{\partial \mathbf{x}_0} \\
-\frac{\partial E_3}{\partial \mathbf{x}_0} &= -\left(\frac{\partial E_3}{\partial \mathbf{x}_4}\right)_{\mathbf{x}_0,\mathbf{x}_5} \frac{\partial \mathbf{x}_4}{\partial \mathbf{x}_0} - \left(\frac{\partial E_3}{\partial \mathbf{x}_5}\right)_{\mathbf{x}_0,\mathbf{x}_4} \frac{\partial \mathbf{x}_5}{\partial \mathbf{x}_0}
\end{aligned}$$

$$\mathbf{f}_0 = -\left(\frac{\partial E_0}{\partial \mathbf{x}_0}\right)_{\mathbf{x}_4,\mathbf{x}_5} + \frac{1}{2}(\mathbf{f}_4)_{\mathbf{x}_0,\mathbf{x}_5} + \frac{1}{2}(\mathbf{f}_5)_{\mathbf{x}_0,\mathbf{x}_4}$$

The intuition behind this result is that each dependent variable (t-vertex) i introduces a $\frac{1}{2}(\mathbf{f}_i)$ term to each of its parents. Derivations for cases with more or less t-vertices and energy constraints defined on edges (bend) proceed similarly.

6 Implicit Model

If the derivation is continued for evaluting the jacobian, a similar pattern emerges: accounting for t-vertex dependencies simply involves splitting the entries in the jacobian evaluated at t-vertices between the parents. However, since the jacobian involves two possibly dependent variables, there may be 0, 2, or 4 parents to split between.

Let $\mathbf{m}$ be the value of the jacobian entry at (u, v) evaluated without accounting for dependencies.

$$\mathbf{m} = \left(\frac{\partial^2 E}{\partial \mathbf{x}_u \partial \mathbf{x}_v}\right) = \left(\frac{\partial \mathbf{f}_u}{\partial \mathbf{x}_v}\right)$$

We split $\mathbf{m}$ in the appropriate jacobian buckets depending on whether or not u and v are t-vertices. The policy basically dictates "split $\mathbf{m}$ between the parents of $x \in \{u, v\}$ if x is a t-vertex."

```
if $u$ is a t-vertex
    if $v$ is a t-vertex
        $\frac{\partial \mathbf{f}_{p1(u)}}{\partial \mathbf{x}_{p1(v)}} {+=} \frac{1}{4}\mathbf{m}$; $\frac{\partial \mathbf{f}_{p1(u)}}{\partial \mathbf{x}_{p2(v)}} {+=} \frac{1}{4}\mathbf{m}$; $\frac{\partial \mathbf{f}_{p2(u)}}{\partial \mathbf{x}_{p1(v)}} {+=} \frac{1}{4}\mathbf{m}$; $\frac{\partial \mathbf{f}_{p2(u)}}{\partial \mathbf{x}_{p2(v)}} {+=} \frac{1}{4}\mathbf{m}$
    else
        $\frac{\partial \mathbf{f}_{p1(u)}}{\partial \mathbf{x}_{v}} {+=} \frac{1}{2}\mathbf{m}$; $\frac{\partial \mathbf{f}_{p2(u)}}{\partial \mathbf{x}_{v}} {+=} \frac{1}{2}\mathbf{m}$
else
    if $v$ is a t-vertex
        $\frac{\partial \mathbf{f}_{u}}{\partial \mathbf{x}_{p1(v)}} {+=} \frac{1}{2}\mathbf{m}$; $\frac{\partial \mathbf{f}_{u}}{\partial \mathbf{x}_{p2(v)}} {+=} \frac{1}{2}\mathbf{m}$
    else
        $\frac{\partial \mathbf{f}_{u}}{\partial \mathbf{x}_{v}} {+=} \mathbf{m}$
```

where $p1(x)$, and $p2(x)$ are the two parent vertices of x.
The evaluation of $\frac{\partial \mathbf{f}_u}{\partial \mathbf{v}_v}$ is identical to above.

7 Results

We tested our force/jacobian propagation method and found that in the case of a fine level mesh with all finer level nodes constrained as t-vertices, the simulation behaved *exactly* the same as a coarser mesh. We give examples of a blanket pinned at two points falling from a horizontal start position. We ran this case with a coarse model, a fine model, and an adaptive model. Our results show that the adaptive model is consistent. In Figure 7, the adaptive results are an average of the low and high resolution models. More nodes are introduced at locations with higher curvature, and more detailed wrinkling behavior is captured. There are no noticable junctions between finer and coarser level elements. Transitions between levels in the animation are devoid of popping effects.

8 Future Work

Our test cases currently do not demonstrate fine wrinkles and hence do not reap a significant performance gain. A more interesting test would be to exercise our model for clothing where wrinkles abound. Our current system does not support self collisions, which limited the cases in which we could test on. We are currently applying this work to model skin, a problem dominated by localized fine wrinkles.

References

[1] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54. ACM Press, 1998.

[2] D. E. Breen, D. H. House, and M. J. Wozny. Predicting the drape of woven cloth using interacting particles. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 365–372. ACM Press, 1994.

[3] K.-J. Choi and H.-S. Ko. Stable but responsive cloth. *ACM Transactions on Graphics (TOG)*, 21(3):604–611, 2002.

[4] Z. D.L. and M. Yuen. Cloth simulation using multilevel meshes. *Computers & Graphics*, 25(3):383–389, June 2001.

[5] E. Grinspun, P. Krysl, and P. Schröder. Charms: a simple framework for adaptive simulation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 281–290. ACM Press, 2002.

[6] D. Hutchinson, M. Preston, and T. Hewitt. Adaptive refinement for mass/spring simulations. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96*, pages 31–45. Springer-Verlag New York, Inc., 1996.

[7] L. Kobbelt. 3 -subdivision. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 103–112. ACM Press/Addison-Wesley Publishing Co., 2000.

[8] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, August 1987.

[9] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. *http://multires.caltech.edu/pubs/diffGeoOps.pdf*, 2002.

[10] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In W. A. Davis and P. Prusinkiewicz, editors, *Graphics Interface '95*, pages 147–154. Canadian Human-Computer Communications Society, 1995.

[11] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. *ACM SIGGRAPH Computer Graphics*, 22(4):269–278, 1988.

[12] J. Villard and H. Borouchaki. Adaptive meshing for cloth animation. In *11th International Meshing Roundtable*, pages 243–252, Ithaca, New York, USA, 15–18 September 2002. Sandia National Laboratories.

[13] V. Volkov and L. Li. Adaptive local refinement and simplification of cloth meshes. In *Proceedings of the first international conference on information technology & applications*, 2002.

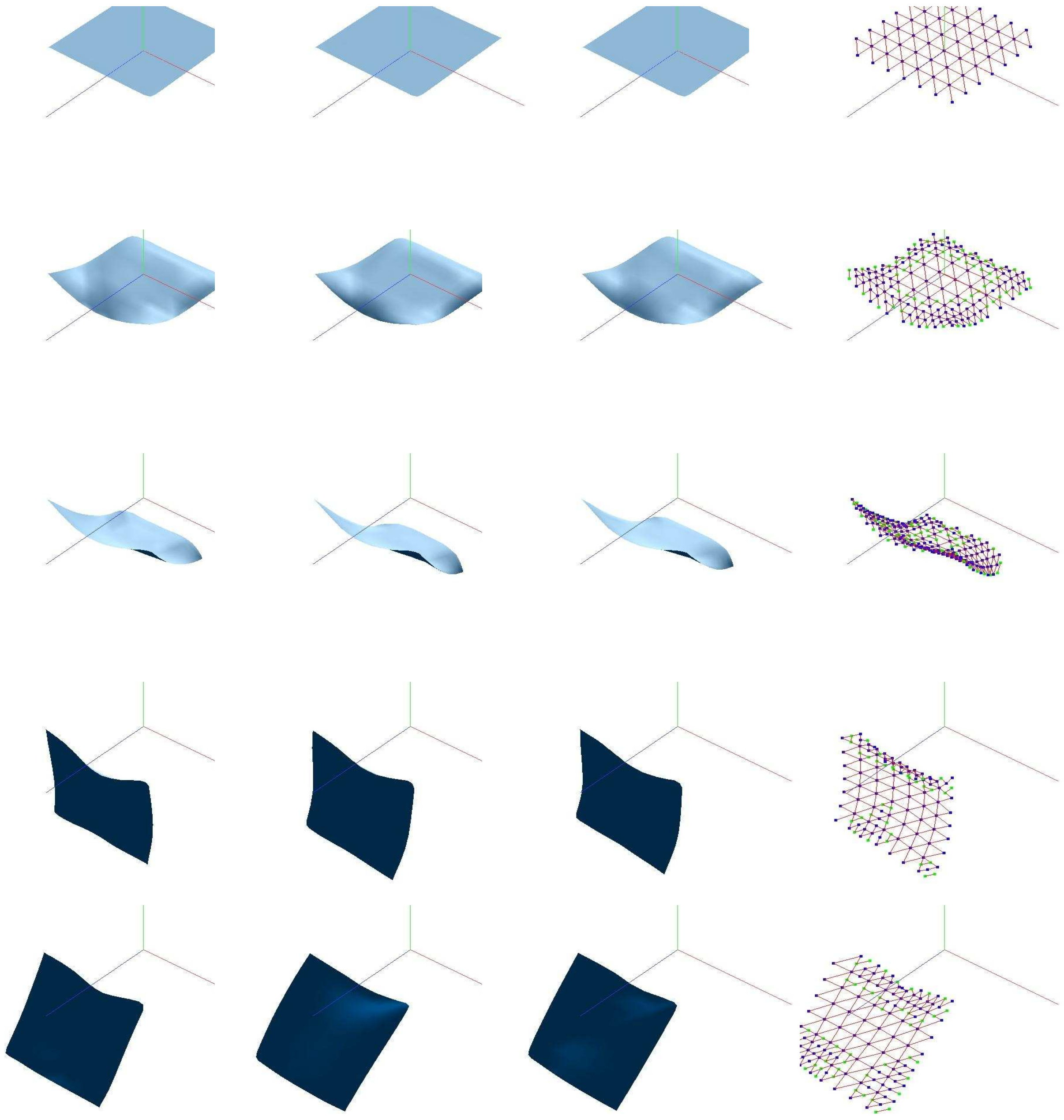

Figure 7: low resolution (a) high resolution (b) and adaptive (c,d) results. Non-t vertices are drawn in blue. T-vertices are drawn in green.