Robust Treatment of Collisions, Contact and Friction for Cloth Animation

Robert Bridson
Stanford University
rbridson@stanford.edu

Ronald Fedkiw
Stanford University
Industrial Light & Magic
fedkiw@cs.stanford.edu

John Anderson
Industrial Light & Magic
janders@pixar.com

Abstract

We present an algorithm to efficiently and robustly process collisions, contact and friction in cloth simulation. It works with any technique for simulating the internal dynamics of the cloth, and allows true modeling of cloth thickness. We also show how our simulation data can be post-processed with a collision-aware subdivision scheme to produce smooth and interference free data for rendering.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling;

Keywords: cloth, collision detection, collision response, contacts, kinetic friction, static friction, physically based animation

1 Introduction

Collisions are a major bottleneck in cloth simulation. Since all points are on the surface, all points may potentially collide with each other and the environment in any given time step. Moreover, for believable animation the number of particles is generally in the tens of thousands or more. Since cloth is very thin, even small interpenetrations can lead to cloth protruding from the wrong side. This is visually disturbing and can be difficult to correct after the fact either in the next time step or in post-processing. While rigid body simulations often have relatively few collisions per object (apart from resting contact), deformable bodies naturally give rise to large numbers of collisions varying in strength from resting contact to high speed impact. Two-dimensional manifolds like cloth are the worst case. Naïve methods for detecting and stopping every collision can quickly grind the simulation to a halt.

This paper presents a collision handling algorithm that works with any method for simulating the internal dynamics (i.e. stretching, shearing, and bending) to efficiently yet robustly produce visually complex motion free from interference as in figure 1. The key idea is to combine a fail-safe geometric collision method with a fast (non-stiff) repulsion force method that models cloth thickness as well as both static and kinetic friction. Ever since [Moore and Wilhelms 1988] proposed that repulsion forces are useful for contact whereas exact impulse-based treatment is useful for high velocity impact, authors have toyed with using both. For example, [Sims 1994] switched between instantaneous impulses for high velocities and penalty spring forces for low velocities to treat his evolving articulated rigid body creatures. Although similar in spirit to our approach, we always use both techniques in a fully hybridized and efficient manner.

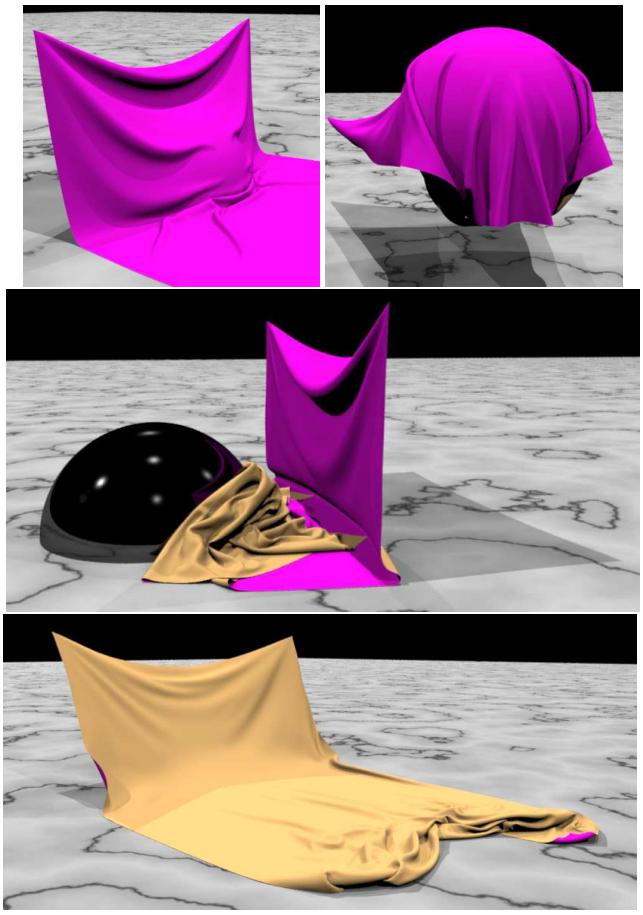

Figure 1: **Initially, a curtain is draped over a ball on the ground. The ball moves, flipping the curtain over on top of itself producing stable folds and wrinkles using our static friction model. Another ball enters the scene and pushes through the complicated structure of folds before slipping underneath unraveling the curtain.**

We view repulsion forces, e.g. during resting contact, as a way to deal with this vast majority of collisions in a simple and efficient manner allowing us to use a more expensive but completely robust method to stop the few that remain. Since our repulsion forces handle most of the self-interaction, it is desirable to make them computationally efficient to apply. Therefore we propose using a repulsion spring model that is not stiff. In contrast, many authors use computationally expensive stiff repulsion springs, e.g. with force inversely

proportional to separation distance, since they do not have a robust alternative for stopping any remaining collisions. See e.g. [Moore and Wilhelms 1988; Lafleur et al. 1991; Baraff and Witkin 1998], although we note that some of the difficulties associated with the stiffness of the repulsion springs was partially alleviated by using an implicit method for the time integration in [Baraff and Witkin 1998].

Our robust geometric collision algorithm is the first scheme that guarantees no dynamic self-interference of cloth. [Moore and Wilhelms 1988] started in this direction proposing a hard-to-solve fifth order polynomial to detect point-face collisions. This was abandoned by the community until [Provot 1997] reduced it to a cubic and extended it to treat edge-edge collisions. However, Provot did not properly account for rounding errors, so self-intersection could still occur. Generally, these difficulties led the community to allow self-intersection, and then attempt to detect and correct it after the fact. For example, [Volino and Magnenat Thalmann 1995; Volino et al. 1995; Volino and Magnenat-Thalmann 1997; Volino et al. 2000] proposed a number of methods such as "most probable" orientations for collisions, i.e. nodes vote on which side they would like to be on and the *majority* wins. Although this gives no guarantee of physical consistency, or that the method even works, they did produce convincing simulations of a ribbon folding, garments crumpling in a dryer, and a stack of cloth. The complexity involved with unraveling self-intersection has led many to use large repulsion forces to keep the cloth well separated, but this leads to visual artifacts with cloth seemingly floating over itself at large distances with little or no friction. Since our method avoids nonphysical self-interference altogether, we do not need large repulsion forces or complicated and unreliable algorithms for detecting and fixing self-intersection. A further advantage over methods that allow self-intersection, even when they succeed in recovering from it, is that our cloth is always properly constrained. This is necessary to capture the bulk and small scale crumpling apparent in complex folds.

A key ingredient of our new algorithm is that we do not work directly with positions but only obtain positions by integrating velocities. Thus, given a current non-interfering state for our cloth, the collision handling problem can be reduced to finding velocities that guarantee a non-interfering state at the end of the time step. Moreover, given a current non-interfering state and a proposed set of new positions at the end of the time step, under a linear trajectory assumption we can compute a velocity to be used along with the initial state in our collision processing algorithm. This allows us to cleanly separate the time evolution of the internal cloth dynamics (and the environment around the cloth) from the collision processing algorithm. That is, the result of the time evolution is merely used as an initial guess for the final position of the cloth. Then this initial guess is modified to account for any collisions. This allows us to easily integrate our collision, contact and friction processing algorithm with a pre-existing cloth dynamics engine.

Our approach to static and kinetic friction is based on the repulsion forces and is trivial to apply even for cloth-cloth interaction. Correctly preventing self-intersection and modeling static and kinetic friction, especially for cloth-cloth contacts, is *essential* for producing the detailed time-evolving folds and wrinkles exhibited by cloth. Our treatment gives highly realistic folds and wrinkles as demonstrated in the figures.

2 Other Work

There is a rich history of research on contact and collisions in the graphics community, and we cannot possibly cover it all due to space limitations. However, we will mention a number of these works where appropriate throughout the text. For example, [Baraff and Witkin 1998; Provot 1995; Provot 1997; Volino et al. 1995; Jimenez and Luciani 1993; Moore and Wilhelms 1988] are cited a number of times.

Baraff carried out a detailed study of numerical methods for rigid body motion with contact and friction in a series of papers [Baraff 1989; Baraff 1990; Baraff 1991; Baraff 1993; Baraff 1994]. [Gourret et al. 1989] simulated a hand grasping an object using a finite element model for both the hand and the object. They detected collision as the overlapping of volumetric objects and subsequently treated collision, contact and friction based on the size of the overlap (including reactive repulsive forces). [Mirtich and Canny 1995] showed that one could produce physically plausible results modeling colliding, rolling, sliding, and resting contact for rigid bodies as a series of collision impulses, or micro-collisions. For example, a block sitting on a table experiences many micro-collisions keeping it from sinking into the table. They used an infinitesimal collision time, Poisson's hypothesis, and a Coulomb friction model. [O'Brien and Hodgins 1999] used a finite element model to simulate elastic brittle objects producing impressive animations of fracture. Collisions were detected via static interpenetration and resolved with penalty forces. Although the penalty forces could be stiff, they stated that their finite element model was relatively just as stiff dictating a small time step anyhow.

Although we use triangles to represent our surface, other representations may be used. [Herzen et al. 1990] addressed collisions between parametric surfaces, [Grinspun and Schröder 2001] worked with subdivision surfaces, and an implicit surface formulation of contact and collision processing was demonstrated in [Gascuel 1993; Desbrun and Gascuel 1994].

3 Cloth Model

Since this paper is concerned with collisions, particularly self-collisions, we do not address internal cloth dynamics. Those interested in cloth modeling are referred to the survey article of [Ng and Grimsdale 1996] and the book of [House and Breen 2000]. We also make specific mention of the CAD apparel system in [Okabe et al. 1992], the work of [Breen et al. 1994] using experimentally determined measurements for cloth properties, and the seminal papers of [Terzopoulos et al. 1987; Terzopoulos and Fleischer 1988a; Terzopoulos and Fleischer 1988b; Terzopoulos and Witkin 1988] on constitutive modeling of deformable bodies for computer graphics. In addition, [Baraff and Witkin 1998] proposed an implicit time stepping method and generated convincing results despite dropping nonsymmetric terms from their matrix. Further approximations were made in [Desbrun et al. 1999], e.g. violating local preservation of angular momentum, in order to obtain interactive rates while sacrificing a little realism.

For the purposes of demonstrating our collision handling, we use a simple mass-spring model for the internal cloth dynamics, as opposed to a true constitutive model. However, in figure 4 we also illustrate the efficacy of our approach with a highly sophisticated model from an industrial production system. In our basic model, particles are arranged in a rectangular array with structural springs connecting immediate neighbors. Diagonal springs provide shear support, and springs connected to every other node (with a stabilization spring attached to the center node in between) resist bending. We make the edges and corners of the cloth slightly heavier by giving all particles the same mass instead of a mass proportional to the area of the surrounding cloth.[1] The heavier edges and corners give the cloth an attractive flare similar to that of real cloth where tailors often make hems a little heavier. This basic cloth model has been used by many authors, e.g. [Provot 1995].

[1]The equal mass assumption also simplifies many of the formulas presented in this paper. Generalizing to the unequal mass case is straightforward.

4 Limiting the Strain and Strain Rate

Sometimes triangles are undesirably stretched or compressed by large percentages. A rule of thumb in computational mechanics is that a triangle edge should not change length by more than 10% in a single time step, see e.g. [Caramana et al. 1998]. This can be enforced by either adaptively decreasing the time step or nonphysically decreasing the strain rate. This rule of thumb is generally used to obtain better accuracy, as opposed to stability, and thus it is used in conjunction with implicit time stepping algorithms as well, see e.g. [Baraff and Witkin 1998].

[Provot 1995] addressed this issue in a novel way processing the cloth after each time step with an iterative algorithm that repairs excessively deformed triangles. This algorithm focused on the overall strain as opposed to the strain rate (although [Provot 1995] mistakenly referred to this as the deformation rate). [Provot 1995] looped through the mesh changing the positions of the nodes on edges that had deformed by over 10%. Since adjusting the position of one node affects the length of all the edges containing it, an iterative process was used. Good results were obtained even though the algorithm does not converge in certain situations, e.g. when all the boundaries of the cloth mesh have constrained (fixed) positions that force an expansion beyond 10%. [Provot 1995] illustrated that this iterative method was significantly more efficient than arbitrarily increasing the spring stiffness when one is dissatisfied with the degree of mesh deformation in a numerical simulation.

Although this method seems to work well, it involves moving nodes and can therefore induce self-intersection in the mesh. Thus, to fit this method into the framework of our collision processing algorithm, we adjust velocities instead of positions. At each time step, we calculate the candidate new spring lengths using the current velocities. Then we apply momentum-conserving corrective impulses to the velocities to ensure that all springs are deformed by a maximum of 10% from their rest length at the end of the time step (ignoring bending springs). These impulses influence the future strains of surrounding springs, and thus an iterative procedure is needed to guarantee that no spring deforms to over 10% of its rest length during the time step. This is essentially equivalent to using biphasic springs with a much stiffer spring constant beyond 10% deformation, and the iterative procedure is similar to using implicit time stepping when the stiffer spring constant is activated. This mimics the physical behavior of cloth (and skin!), which offers little resistance to small deformations but becomes stiff after a critical deformation is reached.

We apply this deformation limiting procedure using a Jacobi iterative approach (parallel rather than sequential), and although convergence is not guaranteed, generally only one or two iterations per time step are required for visually pleasing results. Although a Gauss-Seidel iterative approach (sequential rather than parallel) generally converges faster, it can introduce a noticeable bias according to which parts of the cloth are updated first (although this can be mitigated to some degree by using random orderings). Moreover, Jacobi style iteration is easier to parallelize for high performance.

In addition to the strain, we also limit the strain rate according to the rule of thumb mentioned above. Although, this is usually done by adaptively reducing the time step, these smaller time steps can lead to a loss of efficiency. To avoid slowing the simulation, we continually monitor the strain rate and use momentum-conserving impulses to reduce it so that springs do not change their current length by more than 10% during a time step. This trade-off of accuracy for efficiency does not seem to induce any unwanted visual artifacts and is similar to the traditional damping of an implicit time discretization of the equations. We use a Gauss-Seidel iterative approach in order to accelerate convergence. Only a few iterations are needed as the fastest deforming edges are rapidly damped to reasonable deformation rates. Convergence is not required since we can still adaptively reduce the time step, and after only a few iterations only a moderate reduction of the time step is necessary. [Volino et al. 1995] proposed a philosophically similar technique that monitors local mechanical energy variations and artificially distributes kinetic energy through momentum transfers in regions where instability might occur. Similarly, [Baraff and Witkin 1998] used their implicit time integration scheme to automatically damp the local energy generated by their treatment of collisions.

5 Time Discretization

We cleanly separate the time evolution of the internal cloth dynamics (and the environment around the cloth) from the collision processing algorithm. This allows us to easily integrate our collision, contact and friction processing algorithms with a pre-existing cloth dynamics engine. Starting at time $t = 0$ with cloth positions $\mathbf{x}^0$ and velocities $\mathbf{v}^0$, the algorithm is as follows. For $n = 0, 1, 2, \ldots$

- Select a collision time step size Δt and set $t^{n+1} = t^n + \Delta t$
- Advance to candidate positions $\bar{\mathbf{x}}^{n+1}$ and velocities $\bar{\mathbf{v}}^{n+1}$ at time t^{n+1} with the cloth internal dynamics
- Compute the average velocity $\bar{\mathbf{v}}^{n+1/2} = (\bar{\mathbf{x}}^{n+1} - \mathbf{x}^n)/\Delta t$
- Check $\mathbf{x}^n$ for proximity (section 6), then apply repulsion impulses (section 7.2) and friction (section 7.3) to the average velocity to get $\tilde{\mathbf{v}}^{n+1/2}$
- Check linear trajectories from $\mathbf{x}^n$ with $\tilde{\mathbf{v}}^{n+1/2}$ for collisions (section 6), resolving them with a final midstep velocity $\mathbf{v}^{n+1/2}$ (sections 7.4 and 7.5)
- Compute the final positions $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+1/2}$
- If there were no repulsions or collisions, set $\mathbf{v}^{n+1} = \bar{\mathbf{v}}^{n+1}$
- Otherwise, advance the midstep velocity $\mathbf{v}^{n+1/2}$ to $\mathbf{v}^{n+1}$ (section 7.6)

When repulsions or collisions appear, our method for determining the final velocities is essentially central time differencing [Hughes 1987]. In fact, we use central time differencing for our internal cloth dynamics as well, though we stress that any reasonable algorithm could be used for that purpose, e.g. one large implicit time step as in [Baraff and Witkin 1998] or many small explicit Runge-Kutta steps.

The algorithm is stable for any collision time step Δt, thus Δt can be chosen adaptively in a straightforward manner. For example, we choose a minimum Δt as the time step of the cloth dynamics evolution and a maximum Δt on the order of one frame, and start with the maximum. We halve the time step when an actual collision occurs, i.e. the repulsion forces aren't adequate, and try the time step over again only doing the full collision processing at the minimum Δt. Whenever we get three successful time steps in a row we double Δt again. Adaptive time stepping was also addressed in [Baraff and Witkin 1998].

6 Proximity and Collision Detection

To accelerate the detection of proximities for repulsions and of intersections for collisions, we use an axis-aligned bounding box hierarchy. It is built bottom-up once at the beginning of the simulation using the topology of the cloth mesh. In one sweep we greedily pair off adjacent triangles to get parent nodes, then in a sweep in the opposite direction pair off these to get the next level up, and so on alternating sweep directions until we are left with one root node.

At each time step we calculate the extents of the axis-aligned boxes for the repulsion calculation (section 7.2) by taking a box around each triangle enlarged by the thickness of the cloth (e.g. 10^{-3}m or 1mm), and then taking the union of the extents in each axis direction as we move up the hierarchy. We also recalculate the hierarchy for each iteration of the collision algorithm (section 7.4)

by taking a box around each triangle *and* its candidate position at the end of the step (since we have to cover the path that the triangle takes during the time step) enlarged by the rounding error tolerance (e.g. 10^{-6}m), again merging as we move up the hierarchy. In both cases, we get a set of candidates for interference by intersecting the box to be tested with the root of the tree; only if it overlaps do we recursively check its children, proceeding until we reach the leaves of the tree. The leaves we reach indicate on which triangles the actual geometry tests need to be performed. These tests break down into checking points against triangular faces and edges against other edges (naturally we don't compare a point against the triangle that contains it, or an edge against an edge that shares an endpoint). For more details on hierarchical methods and bounding volume hierarchies, see [Hahn 1988; Webb and Gigante 1992; Barequet et al. 1996; Gottschalk et al. 1996; Lin and Gottschalk 1998]. Further pruning of unnecessary tests between adjacent patches in low curvature regions of cloth is possible, at least for static proximity tests, see [Volino and Magnenat-Thalmann 1994; Volino et al. 1995].

In what follows we use the shorthand $\vec{x}_{ij}$ to mean $\vec{x}_i - \vec{x}_j$.

To check if a point $\vec{x}_4$ is closer than some thickness h to a triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ with normal $\hat{n}$ we first check if the point is close to the plane containing the triangle: $|\vec{x}_{43} \cdot \hat{n}| < h$. If so, we project the point onto the plane and compute the barycentric coordinates w_1, w_2, w_3 with respect to the triangle:

$$\begin{bmatrix} \vec{x}_{13} \cdot \vec{x}_{13} & \vec{x}_{13} \cdot \vec{x}_{23} \\ \vec{x}_{13} \cdot \vec{x}_{23} & \vec{x}_{23} \cdot \vec{x}_{23} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \vec{x}_{13} \cdot \vec{x}_{43} \\ \vec{x}_{23} \cdot \vec{x}_{43} \end{bmatrix}$$

$$w_1 + w_2 + w_3 = 1.$$

These are the normal equations for the least-squares problem of finding the point $w_1\vec{x}_1 + w_2\vec{x}_2 + w_3\vec{x}_3$ (in the plane) closest to $\vec{x}_4$. If the barycentric coordinates are all within the interval $[-\delta, 1+\delta]$ where δ is h divided by a characteristic length of the triangle, the point is close.

To check if an edge $\vec{x}_1\vec{x}_2$ is close to another edge $\vec{x}_3\vec{x}_4$ we find the pair of points, one on each edge, that are closest and check their distance. The search for the two closest points begins by checking for the degenerate case of parallel edges, i.e. if $|\vec{x}_{21} \times \vec{x}_{43}|$ is smaller than a round-off tolerance. If so, it reduces to a simple one-dimensional problem. Otherwise, we find the points on the infinite lines that are closest via the normal equations:

$$\begin{bmatrix} \vec{x}_{21} \cdot \vec{x}_{21} & -\vec{x}_{21} \cdot \vec{x}_{43} \\ -\vec{x}_{21} \cdot \vec{x}_{43} & \vec{x}_{43} \cdot \vec{x}_{43} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \vec{x}_{21} \cdot \vec{x}_{31} \\ -\vec{x}_{43} \cdot \vec{x}_{31} \end{bmatrix}.$$

If these points are on the finite edges we are done, otherwise we clamp them to the endpoints. The point that moved the most during clamping is guaranteed to be one part of the answer, and the second point is found by projecting the first onto the second infinite line and clamping to the finite edge. The direction pointing from one point to the other is saved as the "normal" vector. We also save their relative positions along the edges, i.e. the weights $0 \le a, b \le 1$ so that the points are $\vec{x}_1 + a\vec{x}_{21}$ and $\vec{x}_3 + b\vec{x}_{43}$.

To detect a collision between a moving point and a moving triangle, or between two moving edges, we first find the times at which the four points are coplanar assuming they move with constant velocity over the collision time step as in [Moore and Wilhelms 1988; Provot 1997; Doghri et al. 1998]. The latter two showed this reduces to finding the roots of a cubic equation,

$$(\vec{x}_{21} + t\vec{v}_{21}) \times (\vec{x}_{31} + t\vec{v}_{31}) \cdot (\vec{x}_{41} + t\vec{v}_{41}) = 0.$$

Any roots outside of the interval $[0, \Delta t]$ are discarded, and then the remaining roots are checked in increasing order with proximity tests at time t. If the elements are closer than a small rounding error tolerance (10^{-6}m for our simulations, which is 1000 times smaller than the cloth thickness), we register a collision. We likewise check for proximity at the end of the time step, $t = \Delta t$, in case rounding errors hide a collision at the boundary between two time steps. While earlier works neglected rounding error, our approach guarantees (if collisions are resolved) that the cloth is separated by at least the rounding error tolerance at every time step and never self-intersects during time steps.

7 Contact and Collision Response

7.1 Interpolating within the cloth

We often need to deal with two points from the cloth, computing their relative velocity or applying an impulse to them. However, we cannot directly look up or alter the velocities of such points, and instead must deal with the corners of the triangle or endpoints of the edges.

To compute the velocity of a point interior to a triangle or edge we use linear interpolation, which is exact for affine deformations (i.e. translation, rotation, and affine shearing and scaling). In particular, if a point interior to a triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ has barycentric coordinates w_1, w_2, w_3 (calculated during proximity or collision detection) its interpolated velocity is $w_1\vec{v}_1 + w_2\vec{v}_2 + w_3\vec{v}_3$, and similarly if a point interior to an edge $\vec{x}_1\vec{x}_2$ is the fraction a along the edge then its interpolated velocity is $(1-a)\vec{v}_1 + a\vec{v}_2$. Note that we are finding the velocity of a specific piece of material involved in a contact or collision, i.e. the weights w_i or a are fixed so their time derivatives do not appear.

If an impulse of magnitude I in direction $\hat{n}$ needs to be applied to two points in the cloth (i.e. $I\hat{n}$ to the first and $-I\hat{n}$ to the second), we instead apply impulses to the triangle corners or edge endpoints, weighted by the barycentric coordinates, so that the desired change in relative (interpolated) velocity for the two points is achieved. For the point-triangle case, where an interior point of triangle $\vec{x}_1\vec{x}_2\vec{x}_3$ with barycentric coordinates w_1, w_2, w_3 is interacting with point $\vec{x}_4$, we compute adjusted impulses

$$\tilde{I} = \frac{2I}{1 + w_1^2 + w_2^2 + w_3^2}$$

$$\begin{aligned} \vec{v}_i^{new} &= \vec{v}_i + w_i(\tilde{I}/m)\hat{n} \qquad i = 1,2,3 \\ \vec{v}_4^{new} &= \vec{v}_4 - (\tilde{I}/m)\hat{n} \end{aligned}$$

assuming all nodes have mass m. For the edge-edge case where a point with relative position a along the edge $\vec{x}_1\vec{x}_2$ interacts with a point with relative position b along the edge $\vec{x}_3\vec{x}_4$, the adjusted impulses are

$$\tilde{I} = \frac{2I}{a^2 + (1-a)^2 + b^2 + (1-b)^2}$$

$$\vec{v}_1^{new} = \vec{v}_1 + (1-a)(\tilde{I}/m)\hat{n} \qquad \vec{v}_2^{new} = \vec{v}_2 + a(\tilde{I}/m)\hat{n}$$
$$\vec{v}_3^{new} = \vec{v}_3 - (1-b)(\tilde{I}/m)\hat{n} \qquad \vec{v}_4^{new} = \vec{v}_4 - b(\tilde{I}/m)\hat{n}.$$

Weighting the impulses in this way introduces appropriate torques for off-center interactions as well as giving continuity across triangle boundaries, and converges to the expected formulas when the interior points approach mesh nodes.

7.2 Repulsions

Resolving the tens of thousands of collisions that can readily occur in folding and contact situations can be prohibitively expensive. This is why repulsion forces are mandatory. They dramatically reduce the number of collisions usually eliminating them altogether making our collision processing algorithm not only tractable but efficient. Our cloth is given a realistic thickness, e.g. 1mm, and repulsion forces are only applied between pieces of cloth that have this close proximity. As discussed in section 6, we use an axis-aligned bounding box hierarchy to make this proximity detection

efficient. Proximity is determined for both point-triangle pairs and edge-edge pairs. If a pair is close enough, then two kinds of repulsion forces are applied. The first is based on an inelastic collision, and the second is a spring based force.

[Baraff and Witkin 1992; Baraff and Witkin 1994] discussed collision modeling for deformable bodies pointing out that when a discretized rod collides with a wall, the endpoint of the rod should come impulsively to rest, i.e. the endpoint is subject to a completely inelastic collision impulse. Subsequently the rod stores energy in compression and then expands, separating from the wall. The loading and unloading of the elastic rod models a completely elastic collision. They pointed out that inelasticity can only be introduced by adding damping forces internal to the rod that dissipate energy due to the collision. Other authors have used completely inelastic collisions as well, such as [Desbrun et al. 1999] and [Carignan et al. 1992] (who used completely inelastic collisions to remove the "kicks" generated by the repulsion springs of [Lafleur et al. 1991]). We take a similar approach, modeling all cloth-cloth collisions and cloth-object collisions using an identically zero restitution coefficient in Poisson's hypothesis. The energy stored in deformations of our mass-spring model when one of the nodes abruptly comes to rest against an object in its path is released as the cloth restores itself, causing it to bounce. In fact, most real-world cloth-cloth and cloth-object collisions are fairly inelastic, so even with a zero restitution coefficient one should take care to monitor the energy stored within the cloth. We do this intrinsically by limiting the strain rate as discussed in section 4.

Since our repulsion forces are meant to dramatically reduce the number of subsequent collisions, we incorporate a completely inelastic collision into our repulsion force. Suppose two points in the cloth, one inside a triangle and one a node of the mesh or both interior to mesh edges, are close and have relative velocity v_N in the normal direction which is less than zero, i.e. they are approaching each other. (See section 7.1 above for details on interpolating velocities interior to triangles and edges, and section 6 for details on the normal direction used in point-triangle and edge-edge interactions.) To stop the imminent collision we apply an inelastic impulse of magnitude $I_c = mv_N/2$ in the normal direction. (See section 7.1 for how we distribute the impulse to the mesh nodes involved.)

Since our cloth model includes a realistic cloth thickness, we would like to ensure that pieces of the cloth are well separated at a distance on the order of this cloth thickness. This helps cloth to slide over itself (and objects) without the excessive snagging caused by the discretization errors resulting from the representation of smooth surfaces with discrete triangles. When pieces of cloth are too close together, there is a compression of cloth fibers, and a second repulsion force is applied to model this compression. The repulsion force is proportional to the overlap beyond the cloth thickness h (e.g. 1mm). However, since our robust collision handling algorithm (see section 7.4) will stop every intersection, our spring repulsion force is limited to a maximum when the objects touch, thus avoiding problems with stiffness. Furthermore, we limit our repulsion force so that objects are never propelled outside this overlap region in a single time step. This not only helps to reduce stiffness, but allows cloth in contact to stay close enough together to feel repulsion forces in subsequent time steps. This is important since the repulsion force magnitude is used to model friction, and thus friction forces are also felt in future time steps producing stable folds and wrinkles that add to the visual realism. Many other authors have used spring based repulsion forces, see e.g. [Jimenez and Luciani 1993; Marhefka and Orin 1996], but their methods suffered from undue stiffness since an arbitrary amount of interpenetration was allowed to occur. Again, this is alleviated in our model by the robust geometric collision algorithm that stops interpenetration resulting in a bound on the magnitude of the spring based repulsion force.

The spring based repulsion force is modeled with a spring of stiffness k. As a simple heuristic, we found that matching the stiffness of the stretch springs in the cloth gave good results. The overlap is

$$d = h - (\vec{x}_4 - w_1\vec{x}_1 - w_2\vec{x}_2 - w_3\vec{x}_3)\cdot\hat{n}$$

giving a spring force of kd in the direction $\hat{n}$. Multiplying by Δt gives the impulse. As discussed above, we limit this so that the relative velocity change will reduce the overlap by at most some fixed fraction (e.g. $.1h$) in one Δt time step. If the normal component of relative velocity $v_N \geq .1d/\Delta t$ already we apply no repulsion, otherwise we compute the impulse magnitude

$$I_r = -\min\left(\Delta t k d, m\left(\frac{.1d}{\Delta t} - v_N\right)\right)$$

and distribute it to the mesh nodes as explained in section 7.1.

The repulsion forces can either be applied sequentially or all at once in a parallel update. One drawback of the parallel update is that situations involving multiple interactions can lead to undesirable behavior, e.g. as impulses from multiple inelastic collisions are added together. This can be alleviated to some degree by keeping track of the number of interactions a node is involved in and dividing the resulting impulses by that number. Another remedy consists of multiplying the inelastic collision impulses by a suitable relaxation parameter less than one. We found .25 works fine though the algorithm seems fairly insensitive to small changes in this value.

7.3 Friction

We use Coulomb's model for friction, both static and kinetic, with a single friction parameter μ. The repulsion force F_R from section 7.2 is the negative of the normal force F_N pressing the cloth together. Therefore a friction force, in the direction of the pre-friction relative tangential velocity $\vec{v}_T^{pre}$ but opposite to it, has magnitude at most μF_N. This integrates to an impulse of magnitude $\mu F_N \Delta t$ in the same direction, and thus a change in the relative tangential velocity of at most $\mu F_N \Delta t/m$ where m is the mass (assumed equal for all particles involved). If this is larger than $|\vec{v}_T^{pre}|$, then either the cloth was slipping and stopped due to kinetic friction, or was stuck and shouldn't be allowed to start slipping due to static friction. Either way the new relative tangential velocity should be zero. If not, we can simply subtract this off the magnitude of the relative tangential velocity to account for kinetic friction slowing down the slipping. This calculation can be simplified by noting that $F_N\Delta t/m$ is just Δv_N, the change in relative velocity in the normal direction, which can be directly calculated in the repulsion algorithm. Then the decrease in the magnitude of the relative tangential velocity is $\min(\mu\Delta v_N, |\vec{v}_T^{pre}|)$, i.e. our final relative tangential velocity is

$$\vec{v}_T = \max\left(1 - \mu\frac{\Delta v_N}{|\vec{v}_T^{pre}|}, 0\right)\vec{v}_T^{pre}$$

We apply impulses to achieve this for both point-face proximities and edge-edge proximities.

In effect, we are modeling frictional contact with micro-collisions. We note that if we applied our friction algorithm to rigid bodies, it would solve the inclined plane problem as discussed in [Mirtich and Canny 1995]. A similar algorithm for kinetic friction was proposed by [Jimenez and Luciani 1993] who also calculated a normal force F_N from the magnitude of their spring repulsion force and subsequently used it to evaluate their Coulomb friction model. For static friction they attached a spring between the closest points of two objects when the normal force is nonzero. This attractive spring models static friction until the force exerted by this spring reaches the threshold $\mu_s F_N$. Beyond this, the spring is removed and only kinetic friction applies.

7.4 Geometric Collisions

Repulsion forces alone cannot ensure that no interpenetrations will occur since positions are only checked at discrete moments in time. For robust collision handling, the path of the cloth between time steps must be considered as discussed in section 6.

Some authors back up simulations in time to treat collisions in chronological order, e.g. [Hahn 1988]. When a single time step may have thousands of collisions and contacts, as is characteristic of highly deformable bodies like cloth, this is quite expensive and can grind the simulation to a halt. The problem was addressed for rigid bodies by [Mirtich 2000] who processed the rigid bodies in parallel using a timewarp algorithm to back up just the objects that are involved in collisions while still evolving non-colliding objects forward in time. This method works well except when there are a small number of contact groups which unfortunately is the case for cloth as the entire piece of cloth has every node in contact with every other node through the mass-spring network.

Instead of rewinding the simulation to process one collision at a time, we resolve them all simultaneously with an iterative procedure as did [Volino et al. 1995; Provot 1997; Milenkovic and Schmidt 2001]. This does not give the same result as processing the collisions in chronological order. However, there is enough uncertainty in the collision modeling and error in the cloth discretization that we are already guaranteed to not get *the* exact physically correct answer. Instead we will obtain *a* physically plausible solution, i.e. one of many possible physically correct outcomes which may vary significantly with slight perturbations in initial conditions or the inclusion of unmodeled phenomena such as interactions between fuzzy strands of cloth. More details on sampling plausible solutions according to probability distributions reflecting a number of factors can be found in [Chenney and Forsyth 2000] who addressed the related problem of multiple colliding rigid bodies. Simulating plausible motion in chaotic scenarios was also addressed by [Milenkovic and Schmidt 2001] who studied problems where large numbers of rigid bodies were in a single contact group and employed iterative collision processing techniques, phrased as optimization procedures, in order to adjust the positions of the bodies to avoid overlap.

As discussed in section 6, our geometric collision processing algorithm is activated either when a collision actually occurs or when geometry (points and faces or edges and edges) is in (too) close proximity at the end of a time step. Thus, we need to account for both approaching and separating objects when a "collision" is registered. If the geometry is approaching, we apply a completely inelastic repulsion impulse. Otherwise, if the geometry is already separating (as may happen at the end of the time step, i.e. a close call rather than a true collision), we apply a spring based repulsion force. See section 7.2 for more details on both of these.

The collision impulses can either be applied sequentially or all at once in a parallel update. Once again, in the case of a parallel update, situations involving multiple interactions can lead to undesirable behavior. This can once again (similar to the repulsion forces) be alleviated by dividing the resulting impulses by the number of interactions, or by using a suitable relaxation parameter less than one (e.g. .25).

While processing all the collisions that occurred during a time step, we may have inadvertently caused new secondary collisions to occur. In order to keep the cloth interference free, we must analyze the newly computed post-collision path of the cloth for possible collisions and process these as well. This means that the bounding box hierarchy needs to be adjusted to account for the new post-collision velocities. Then secondary collisions can be detected and corrected, etc., and the process continues until a final interference free state can be computed. Since relatively large bounding boxes that contain the moving triangles need to be recomputed for every iteration, and a cubic equation has to be solved for every possible collision, this may be expensive. Luckily, our repulsion forces tend to catch and treat almost all collisions making the iteration scheme here practical to apply even for high velocity cloth with many nodes and a high degree of folding and contact. Also, there are some multiple collision situations, such as a node sandwiched between two approaching triangles, that are resolved immediately if we apply impulses in parallel (but can iterate for a long time if they are applied sequentially instead, although Gauss-Seidel generally converges faster than Jacobi iteration). However, there are still situations where many iterations are required, so after a few iterations we switch to a failsafe method which quickly eliminates all collisions. We use the method proposed by [Provot 1997], but not followed through in the literature, possibly because the formulas proposed in [Provot 1997] *do not* give true rigid body motion. We give corrected versions below.

7.5 Rigid Impact Zones

[Provot 1997] proposed collecting the nodes involved in multiple collisions into "impact zones" which are treated as rigid bodies. This is justified by observing that when cloth bunches together friction will prevent most relative motion. Thus, after a few iterations of applying local impulses as outlined above, we instead switch to merging lists of nodes representing impact zones. Initially, each node in its own list. Then, when a point-face or edge-edge collision occurs, the lists containing the four involved nodes are merged together into one larger impact zone. The impact zones are grown until the cloth is collision free. The velocity of the nodes in the impact zone is derived from a rigid body motion that preserves linear and angular momentum. The formula for angular velocity given in [Provot 1997] is flawed, so we present a corrected version here.

To find the rigid body motion we first compute the initial center of mass of the impact zone and its average velocity

$$\vec{x}_{CM} = \frac{\sum_i m\vec{x}_i^n}{\sum_i m}, \qquad \vec{v}_{CM} = \frac{\sum_i m\vec{v}_i^{n+1/2}}{\sum_i m}$$

then the angular momentum of the nodes with respect to their center of mass

$$\vec{L} = \sum_i m(\vec{x}_i^n - \vec{x}_{CM}) \times (\vec{v}_i^{n+1/2} - \vec{v}_{CM})$$

and the 3×3 inertia tensor of the current configuration of nodes (using δ to represent the identity tensor)

$$I = \sum_i m \left(|\vec{x}_i^n - \vec{x}_{CM}|^2 \delta - (\vec{x}_i^n - \vec{x}_{CM}) \otimes (\vec{x}_i^n - \vec{x}_{CM}) \right).$$

The rigid body angular velocity that would preserve angular momentum is $\vec{\omega} = I^{-1}\vec{L}$, so the new instantaneous velocity of node i is

$$\vec{v}_{CM} + \vec{\omega} \times (\vec{x}_i - \vec{x}_{CM})$$

However, we want the average velocity over the time step of finite size Δt, so that the update $\vec{x}^{n+1} = \vec{x}^n + \Delta t \vec{v}^{n+1/2}$ *exactly* corresponds to a rigid body motion, i.e. so that lengths and angles stay fixed. If this last condition is not enforced (it was not addressed in [Provot 1997]), then self-intersection can occur. Assuming that we can accept a small $O(\Delta t)$ error in the axis and angle of the total rotation, we make the approximation that $\vec{\omega}$ stays constant over the time step. Then we find the fixed and rotating components of the position

$$\vec{x}_F = (\vec{x}_i - \vec{x}_{CM}) \cdot \frac{\vec{\omega}}{|\vec{\omega}|} \frac{\vec{\omega}}{|\vec{\omega}|}, \qquad \vec{x}_R = \vec{x}_i - \vec{x}_{CM} - \vec{x}_F$$

giving the final position

$$\vec{x}_i^{n+1} = \vec{x}_{CM} + \Delta t \vec{v}_{CM} + \vec{x}_F + \cos(\Delta t|\vec{\omega}|)\vec{x}_R + \sin(\Delta t|\vec{\omega}|)\frac{\vec{\omega}}{|\vec{\omega}|} \times \vec{x}_R.$$

The new average velocity is then $\bar{v}_i^{n+1/2} = (\bar{x}_i^{n+1} - \bar{x}_i^n)/\Delta t$.

Applied on its own, this impact zone method has a tendency to freeze the cloth into nonphysical clumps. However, a combination of our repulsion forces and the initial collision impulses tend to keep these impact zones small, isolated and infrequent. Moreover, once formed, they are short-lived as the repulsion forces tend to quickly separate the offending nodes.

7.6 Updating the Final Velocity

When there are repulsions or collisions, we need to update the velocity from $\mathbf{v}^{n+1/2}$ to $\mathbf{v}^{n+1}$. For central differencing we need to solve the implicit equation

$$\mathbf{v}^{n+1} = \mathbf{v}^{n+1/2} + \tfrac{\Delta t}{2}\mathbf{a}(t^{n+1}, \mathbf{x}^{n+1}, \mathbf{v}^{n+1})$$

where $\mathbf{a}(t, \mathbf{x}, \mathbf{v})$ is the acceleration.

In many cloth models, such as ours, the damping forces (hence accelerations) are linear in the velocities giving the linear system

$$(I - \tfrac{\Delta t}{2}\tfrac{\partial \mathbf{a}}{\partial \mathbf{v}})\mathbf{v}^{n+1} = \mathbf{v}^{n+1/2} + \tfrac{\Delta t}{2}\mathbf{a}_e(t^{n+1}, \mathbf{x}^{n+1}).$$

Here $\partial \mathbf{a}/\partial \mathbf{v}$ is the Jacobian matrix of accelerations with respect to velocities, and $\mathbf{a}_e(t, \mathbf{x})$ is the elastic component of acceleration, i.e. everything apart from damping. In any reasonable cloth model this matrix will be symmetric positive definite (or can be safely made so, see [Baraff and Witkin 1998]) after multiplying both sides by the mass matrix, i.e. converting velocities into momenta and accelerations into forces. Then the conjugate gradient algorithm (e.g. [Saad 1996]) can be used to solve for $\mathbf{v}^{n+1}$. For nonlinear damping Newton's method can be used requiring similar linear solves.

As an alternative for the nonlinear case, or when solving the linear system proves too difficult, one can explicitly march the velocity forward in time. Starting with $\mathbf{u}_0 = \mathbf{v}^{n+1/2}$ we advance

$$\mathbf{u}_{m+1} = \mathbf{u}_m + k\mathbf{a}(t^{n+1}, \mathbf{x}^{n+1}, \mathbf{u}_{m+1})$$

ending with $\mathbf{v}^{n+1} = \mathbf{u}_M$ where $\Delta t/2 = Mk$. Each substep is solved with fixed point iteration, starting with initial guess $\mathbf{u}_{m+1}^{(0)} = \mathbf{u}_m$ for $\mathbf{u}_{m+1}$ and continuing with

$$\mathbf{u}_{m+1}^{(i+1)} = \mathbf{u}_m + k\mathbf{a}(t^{n+1}, \mathbf{x}^{n+1}, \mathbf{u}_{m+1}^{(i)})$$

for a given number of iterations. The substep size k is chosen with $k|\lambda_{max}(\partial \mathbf{a}/\partial \mathbf{v})| < 1$ in order to guarantee convergence. This is essentially the same as the stability limit of forward Euler time stepping. Note that this can be made more efficient by separating the calculation of acceleration into the elastic component $\mathbf{a}_e$ mentioned above (which does not change) and the damping component.

As mentioned in section 4, we monitor the strain rate in the cloth and introduce additional damping if necessary. This is important for dealing with especially difficult collision situations. While our algorithm can fully robustly handle them, bad strain rates in the cloth due to the collision impulses can cause additional unwanted collisions in subsequent time steps slowing the simulation down and producing visually inaccurate results. In addition, when one node collides and impulsively changes its velocity so that it is dramatically out of sync with surrounding nodes, our strain rate limiting procedure puts the nodes back in sync reducing the velocity of surrounding nodes even though they have not yet been involved in a collision. This keeps our cloth from experiencing unnecessary stress and also increases the likelihood that repulsion forces will stop the surrounding nodes before they actually collide, again dramatically reducing the number of collisions that have to be dealt with.

8 Post-Processing with Subdivision

Sharp folds and wrinkles in the cloth mesh give undesirable angular features when rendered as plain triangles. For visually pleasing animations a smoother surface is desired. Some authors have directly simulated smooth surfaces instead of simple triangle meshes. For example, [Thingvold and Cohen 1992] used dynamic B-splines which allowed them to interactively refine the mesh in regions of interest associating the control points with their dynamic simulation mesh nodes. They derived a number of rules for when, where and how to refine, even detecting when mesh refinement would cause intersection and then either stopping refinement or backing up the simulation in time to avoid intersection. [DeRose et al. 1998] exploited the convex hull properties of their subdivision surface model of cloth to accelerate collision detection. [Grinspun and Schröder 2001] rigorously modeled thin manifolds with subdivision surfaces detecting collisions with their derived bounds on surface normals and refining the mesh as required to resolve them.

We propose a fast and simple yet collision-aware post-processing subdivision scheme to smooth our triangle mesh. Our post-processing scheme takes the existing intersection free simulation data and produces a finer, more detailed and smoother approximation to the manifold. We never have to back the mesh up in time or cease refinement, or even have to consider refinement at all in the simulation. Our algorithm efficiently works independently from the simulation on the positions recorded for each frame. In addition, each frame can be processed independently just as in a rendering pipeline.

Another motivation for our post-processing is found in [Howlett and Hewitt 1998] who addressed cloth collisions with volumetric objects. They ensured that cloth nodes remained outside the objects making the collision-handling algorithm faster and simpler, but allowed edge and face collisions with the objects. These were handled in a post-processing step before rendering where they added nonactive points and adjusted their positions to eliminate intersections. Although we treat cloth-object collisions in a more detailed manner, our subdivision approach naturally allows this clever "lie about it" strategy explained in [Baraff 2001] where small interpenetrations are allowed in the simulation but are corrected before rendering. [Howlett and Hewitt 1998] further processed their cloth in an attempt to preserve area, but we do not undertake this endeavor since the idea of subdivision is that it recovers the *true* geometry approximated by the coarse simulation mesh.

Our post-processing algorithm proceeds as follows. If we allowed intersection with objects in the simulation, we begin by adjusting the cloth positions in the given frame to eliminate them iterating back and forth with an adjustment to eliminate cloth-cloth intersections that those adjustments may have caused. We use the repulsions and collision impulses from sections 7.2 and 7.4. When this is finished, our original mesh is intersection free even accounting for rounding error. We then subdivide the mesh putting a new node at the midpoint of each edge. Since the original mesh was intersection free and the subdivided mesh lies exactly within the original mesh, the subdivided mesh is guaranteed to be intersection free as well.

Next we use the modified Loop subdivision scheme [Loop 2001] to find smoother positions for all the nodes of the subdivided mesh. Unfortunately, moving to these smoothed positions may create intersections. However, we can view the vector from a node's original position to its smoothed position as a pseudo-velocity and apply our collision detection algorithms from section 6 to determine when the intersection would occur. We stop the nodes at that point (or just before that point to avoid difficulties with rounding error) as if they had inelastically collided. We of course need to check again to see if these adjustments to the smoothed positions caused new intersections. Typically only a few iterations are required to eliminate all

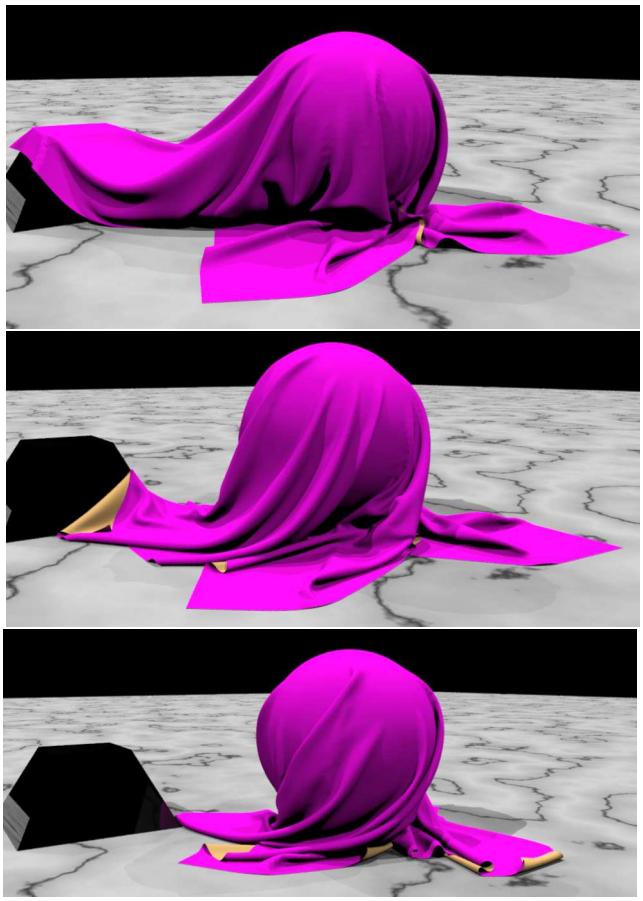

Figure 2: **The friction between a rotating sphere and a piece of cloth draped over it creates a complex structure of wrinkles and folds.**

intersections especially since the convex-hull property of the subdivision means intersections are unlikely in the first place. A solution is guaranteed to exist, since the new nodes can simply be left on the triangle they were created on. Once we have a smoothed but intersection free subdivided mesh, we can subdivide again continuing until the desired resolution is reached. Since the cloth is originally separated by a finite distance, but each step of subdivision smoothing moves the nodes exponentially less and less, we very quickly find no more adjustments need to be made.

We caution the reader that this post-processing technique performs exceptionally well because we use a fairly high resolution dynamic simulation mesh. The efficiency of our repulsion and collision processing algorithms allows the use of such a mesh, and we have not noticed any problems with visual artifacts. However, on a relatively coarser mesh, one should be aware of potential artifacts such as ”popping” that might result from using this subdivision scheme.

9 Examples

We demonstrate several examples using our simple cloth model with highly complicated folding where most of the nodes (tens of thousands in the dynamics and hundreds of thousands after subdivision) are in close contact with each other as opposed to, say, the simple draping of a skirt about a mannequin. In figure 1, a curtain is draped over the ground and a sphere. Our biphasic spring model enables complex wrinkling and eliminates undue deformation. When the sphere moves up and away, the curtain flips back over on itself resulting in a large number of contacts and collisions. The highly

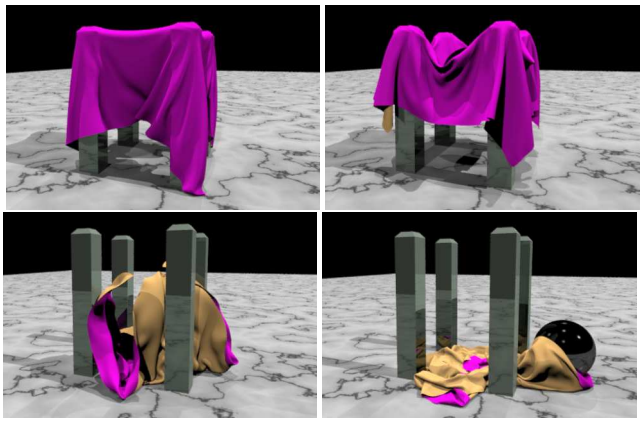

Figure 3: **A tablecloth draped over table legs with no tabletop is dragged to the ground by a descending sphere.**

Not available in this version

Figure 4: **Frames from a production animation of a robe draped over a digital character.**

complex structure of folds and wrinkles is stable due to our static friction model. When a second ball pushes through the complex structure eventually slipping underneath, the algorithm still efficiently and correctly resolves all contacts and collisions. Note how realistically the cloth unravels by the final frame.

Figure 2 illustrates our static and kinetic friction algorithm with a piece of cloth draped over a rotating sphere. Figure 3 shows a tablecloth draped over four table legs with no tabletop. The object-cloth contact is tricky due to the sharp corners of the legs particularly when a sphere descends through the cloth down onto the ground, but our repulsion forces prevent unnatural snagging. Simulation times were reasonable even for these complex examples. Typically, a piece of cloth with 150×150 nodes runs at about 2 minutes per frame on a 1.2GHz Pentium III. Finally, figure 4 shows the draping and folding of a robe around a digital character from a production animation system utilizing a number of our techniques.

10 Conclusions and Future Work

The synergy of efficient repulsion forces combined with robust geometric treatment of collisions has allowed us to efficiently simulate complex cloth motion. The prevention of self-intersection together with kinetic and static friction produces complex, yet stable folding and wrinkling unachievable by simpler approaches. In addition, our post-processing subdivides simulation data without introducing self-intersection resulting in even higher quality animations. Our algorithm makes few assumptions about the internal cloth dynamics, and thus can easily be incorporated into existing codes with advanced cloth models.

We are close to a fully parallel implementation exploiting the parallel nature of most of our scheme. Other areas we plan to develop include modeling different values for kinetic and static friction coefficients, adaptive meshing to better resolve folds, and optimization of the bounding volume hierarchy. Furthermore, we are eager to apply our techniques to characters with highly wrinkled loose fitting skin.

Two rather important problems that we have not addressed are the interactions between cloth with sharp objects and the behavior of cloth when trapped in between two solid deformable or rigid bodies. We refer the reader interested in sharp objects to the recent developments of [Kane et al. 1999; Pandolfi et al. 2002]. For the case of intersecting collision bodies additional technologies like the

ones developed by Baraff, Witkin and Kass (Personal Communication 2002) are required.

11 Acknowledgement

Research supported in part by an ONR YIP and PECASE award N00014-01-1-0620. The first author was supported in part by a Stanford Graduate Fellowship.

The authors would like to thank Igor Neverov, Joseph Teran, and Neil Molino for their help producing the final animations, and Henrik Wann Jensen for the use of his *dali* rendering system.

The authors would like to thank Cliff Plumer, Andy Hendrickson, Sebastian Marino and Lucasfilm for their guidance and support, as well as the images in figure 4.

References

BARAFF, D., AND WITKIN, A. 1992. Dynamic simulation of non-penetrating flexible bodies. In *Proc. of SIGGRAPH 1992*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 303–308.

BARAFF, D., AND WITKIN, A. 1994. Global methods for simulating contacting flexible bodies. In *Computer Animation Proc.*, Springer-Verlag, 1–12.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. In *Proc. of SIGGRAPH 1998*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 1–12.

BARAFF, D. 1989. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *Proc. of SIGGRAPH 1989*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc.

BARAFF, D. 1990. Curved surfaces and coherence for non-penetrating rigid body simulation. In *Proc. of SIGGRAPH 1990*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc.

BARAFF, D. 1991. Coping with friction for non-penetrating rigid body simulation. In *Proc. of SIGGRAPH 1991*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 31–40.

BARAFF, D. 1993. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10, 292–352.

BARAFF, D. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *Proc. of SIGGRAPH 1994*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc.

BARAFF, D. 2001. Collision and contact. In *SIGGRAPH 2001 Course Notes*, ACM.

BAREQUET, G., CHAZELLE, B., GUIBAS, L., MITCHELL, J., AND TAL, A. 1996. BOXTREE: A hierarchical representation for surfaces in 3D. *Comp. Graphics Forum 15*, 3, 387–396.

BREEN, D. E., HOUSE, D. H., AND WOZNY, M. J. 1994. Predicting the drape of woven cloth using interacting particles. In *Proc. of SIGGRAPH 1994*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 365–372.

CARAMANA, E., BURTON, D., SHASHKOV, M., AND WHALEN, P. 1998. The construction of compatible hydrodynamics algorithms utilizing conservation of total energy. *Journal of Computational Physics 146*, 227–262.

CARIGNAN, M., YANG, Y., MAGNENAT-THALMANN, N., AND THALMANN, D. 1992. Dressing animated synthetic actors with complex deformable clothes. In *Proc. SIGGRAPH 1992*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 99–104.

CHENNEY, S., AND FORSYTH, D. A. 2000. Sampling plausible solutions to multi-body constraint problems. In *SIGGRAPH 2000*, ACM Press / ACM SIGGRAPH, Comp. Graphics Proc.

DEROSE, T., KASS, M., AND TRUONG, T. 1998. Subdivision surfaces in character animation. In *Proc. SIGGRAPH 1998*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 85–94.

DESBRUN, M., AND GASCUEL, M.-P. 1994. Highly deformable material for animation and collision processing. In *5th Eurographics worshop on animation and simulation*.

DESBRUN, M., SCHRÖDER, P., AND BARR, A. 1999. Interactive animation of structured deformable objects. In *Graphics Interface*, 1–8.

DOGHRI, I., MULLER, A., AND TAYLOR, R. L. 1998. A general three-dimensional contact procedure for finite element codes. *Engineering Computations 15*, 2, 233–259.

GASCUEL, M.-P. 1993. An implicit formulation for precise contact modeling between flexible solids. In *SIGGRAPH 1993*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 313–320.

GOTTSCHALK, S., LIN, M. C., AND MANOCHA, D. 1996. Obbtree: a hierarchical structure for rapid interference detection. In *Proc. of SIGGRAPH 1996*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 171–179.

GOURRET, J.-P., MAGNENAT-THALMANN, N., AND THALMANN, D. 1989. Simulation of object and human skin deformations in a grasping task. In *Proc. of SIGGRAPH 1989*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 21–30.

GRINSPUN, E., AND SCHRÖDER, P. 2001. Normal bounds for subdivision-surface interference detection. In *Proc. of IEEE Scientific Visualization*, IEEE.

HAHN, J. K. 1988. Realistic animation of rigid bodies. In *Proc. of SIGGRAPH 1988*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc.

HERZEN, B. V., BARR, A. H., AND ZATZ, H. R. 1990. Geometric collisions for time-dependent parametric surfaces. In *Proc. of SIGGRAPH 1990*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 39–48.

HOUSE, D. H., AND BREEN, D. E., Eds. 2000. *Cloth modeling and animation*. A. K. Peters.

HOWLETT, P., AND HEWITT, W. T. 1998. Mass-spring simulation using adaptive non-active points. In *Computer Graphics Forum*, vol. 17, 345–354.

HUGHES, T. J. R. 1987. *The finite element method: linear static and dynamic finite element analysis*. Prentice Hall.

JIMENEZ, S., AND LUCIANI, A. 1993. Animation of interacting objects with collisions and prolonged contacts. In *Modeling in computer graphics—methods and applications*, Springer-Verlag, B. Falcidieno and T. L. Kunii, Eds., Proc. of the IFIP WG 5.10 Working Conference, 129–141.

KANE, C., REPETTO, E., ORTIZ, M., AND MARSDEN, J. 1999. Finite element analysis of nonsmooth contact. *Comput. Methods Appl. Mech. Eng. 180*, 1–26.

LAFLEUR, B., MAGNENAT-THALMANN, N., AND THALMANN, D. 1991. Cloth animation with self-collision detection. In *Proc. of the Conf. on Modeling in Comp. Graphics*, Springer, 179–187.

LIN, M., AND GOTTSCHALK, S. 1998. Collision detection between geometric models: A survey. In *Proc. of IMA Conf. on Mathematics of Surfaces*.

LOOP, C. 2001. Triangle mesh subdivision with bounded curvature and the convex hull property. Tech. Rep. MSR-TR-2001-24, Microsoft Research.

MARHEFKA, D. W., AND ORIN, D. E. 1996. Simulation of contact using a nonlinear damping model. In *Proc. of the 1996 IEEE Int'l Conf. on Robotics and Automation*, IEEE, 1662–1668.

MILENKOVIC, V. J., AND SCHMIDT, H. 2001. Optimization-based animation. In *Proc. of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc.

MIRTICH, B., AND CANNY, J. 1995. Impulse-based simulation of rigid bodies. In *Proc. of 1995 symposium on interactive 3d graphics*, 181–188, 217.

MIRTICH, B. 2000. Timewarp rigid body simulation. In *Proc. of SIGGRAPH 2000*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 193–200.

MOORE, M., AND WILHELMS, J. 1988. Collision detection and response for computer animation. In *SIGGRAPH 1988*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 289–298.

NG, H. N., AND GRIMSDALE, R. L. 1996. Computer graphics techniques for modeling cloth. *IEEE Computer Graphics and Applications*, 28–41.

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *SIGGRAPH 1999*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 137–146.

OKABE, H., IMAOKA, H., TOMIHA, T., AND NIWAYA, H. 1992. Three dimensional apparel CAD system. In *SIGGRAPH 1992*, ACM Press/ACM SIGGRAPH, Comp. Graphics Proc., 105–110.

PANDOLFI, A., KANE, C., MARSDEN, J., AND ORTIZ, M. 2002. Time-discretized variational formulation of non-smooth frictional contact. *Int. J. Num. Methods in Eng. 53*, 1801–1829.

PROVOT, X. 1995. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface*, 147–154.

PROVOT, X. 1997. Collision and self-collision handling in cloth model dedicated to design garment. *Graphics Interface*, 177–89.

SAAD, Y. 1996. *Iterative methods for sparse linear systems*. PWS Publishing. New York, NY.

SIMS, K. 1994. Evolving virtual creatures. In *SIGGRAPH 1994*, ACM Press / ACM SIGGRAPH, Comp. Graphics Proc., 15–22.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Deformable models. *The Visual Computer*, 4, 306–331.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. In *Proc. of SIGGRAPH 1988*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 269–278.

TERZOPOULOS, D., AND WITKIN, A. 1988. Physically based models with rigid and deformable components. In *Graphics Interface*, 146–154.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *SIGGRAPH 1987*, ACM Press/ACM SIGGRAPH, Comp. Graphics Proc., 205–214.

THINGVOLD, J. A., AND COHEN, E. 1992. Physical modeling with B-spline surfaces for interactive design and animation. In *Proc. of SIGGRAPH 1992*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc., 129–137.

VOLINO, P., AND MAGNENAT-THALMANN, N. 1994. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. In *Proc. of Eurographics*, vol. 13 of *Computer Graphics Forum*, Eurographics Association, C–155–166.

VOLINO, P., AND MAGNENAT THALMANN, N. 1995. Collision and self-collision detection: Efficient and robust solutions for highly deformable surfaces. In *Comp. Anim. and Simulation*, Springer-Verlag, D. Terzopoulos and D. Thalmann, Eds., 55–65.

VOLINO, P., AND MAGNENAT-THALMANN, N. 1997. Developing simulation techniques for an interactive clothing system. In *Proc. of the 1997 International Conf. on Virtual Systems and MultiMedia*, IEEE, 109–118.

VOLINO, P., COURCHESNE, M., AND MAGNENAT-THALMANN, N. 1995. Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proc. of SIGGRAPH 1995*, ACM Press / ACM SIGGRAPH, Comput. Graphics Proc.

VOLINO, P., COURCHESNE, M., AND MAGNENAT-THALMANN, N. 2000. Accurate collision response on polygonal meshes. In *Proc. of Computer Graphics*, 179–188.

WEBB, R., AND GIGANTE, M. 1992. Using dynamic bounding volume hierarchies to improve efficiency of rigid body simulations. In *Comm. with Virtual Worlds*, CGI Proc. 1992, 825–841.