Robust High-Resolution Cloth Using Parallelism, History-Based Collisions and Accurate Friction

Andrew Selle, Jonathan Su, Geoffrey Irving, Ronald Fedkiw

Abstract—In this paper we simulate high resolution cloth consisting of up to 2 million triangles which allows us to achieve highly detailed folds and wrinkles. Since the level of detail is also influenced by object collision and self collision, we propose a more accurate model for cloth-object friction. We also propose a robust history-based repulsion/collision framework where repulsions are treated accurately and efficiently on a per time step basis. Distributed memory parallelism is used for both time evolution and collisions and we specifically address Gauss-Seidel ordering of repulsion/collision response. This algorithm is demonstrated by several high-resolution and high-fidelity simulations.

Index Terms—Computer Graphics, Geometric algorithms, Physically based modelling, Cloth, Collision response, Friction

1 Introduction

Cloth simulation is pervasive in film, e.g. the untangling strategies for Monsters Inc. [1], the collision and stiction methods for Terminator 3 and Harry Potter [2], and the wrinkle system for Shrek 2 [3]. Cloth simulation also promises to have significant future impact on the clothing industry [4] (see also [5]). While, some researchers have focused on real-time simulation for computer games or non-hero characters [6] that have lower quality requirements, our focus is on hero characters, online shopping, and related applications that need photorealistic cloth and clothing. Achieving such realism requires higher resolution because the number of bends and folds is limited by the underlying cloth discretization.

Thus the focus of this paper is introducing an approach allowing simulation of extremely high resolution meshes and producing interactions commensurate with this level of detail. This contrasts with previous cloth papers that simulated relatively few triangles: [7] used 10-40 thousand elements, [8] used 5-38 thousand elements, and [9] mostly considered a few thousand elements but their highest resolution simulation was a very thin ribbon with 80 thousand elements that exhibits bending but no folds or wrinkles. These resolutions cannot resolve or simulate folds and wrinkles at the granularity of Figure 2. Most simulation techniques would fail if resolution were increased because of two problems: robustness and tractability. These problems typically manifest themselves in time integration and self-collisions/contact (see Figure 3).

The first contribution of this paper is a modified time integration scheme which is tractable for high-resolution cloth simulations. Time integration for cloth has been studied extensively, and researchers have settled on using either semi-implicit approaches such as [2], [11] or fully implicit approaches [12] as they remove the quadratic time step restriction due to the damping terms.

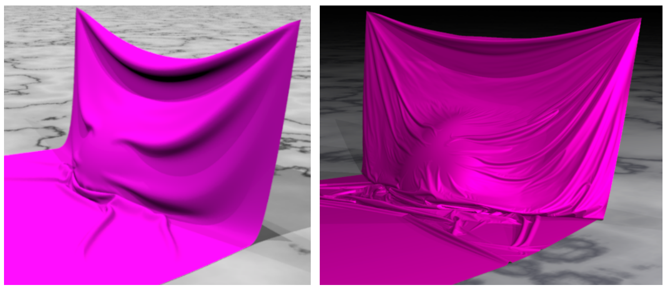


Fig. 1. High resolutions are necessary to represent and simulate highly detailed, so our goal is to maintain the same robustness and quality while having better scalability.

Though fully implicit methods have no time step restriction, accurate time integration requires increasingly small time steps as resolution increases, meaning high resolution simulations are still intractable even with implicit integration schemes. To make our high resolution simulations tractable, we turn to distributed memory parallelism to improve performance. While there has been much work on shared and distributed memory parallelism for cloth simulation [13]–[18], we emphasize the importance of our method's parallel Gauss-Seidel collision/repulsion response.

The second contribution of this paper is to maintain the robustness of the collision algorithm while making the collisions tractable for high-resolution simulations. Self collisions and interactions are important not only because they prevent interpenetration but because they also force the cloth to form folds and wrinkles. Recent works have demonstrated that it is possible to stop all collisions even in complex scenarios [10], [19]–[21], while other works have shown that untangling is useful as well [9], [22], [23] especially in situations such as pinching

Fig. 2. Two examples of cloth showing the types of intricate folds and wrinkles we wish to simulate; both would require high resolution simulation. A satin bed sheet (left) shows that larger scale cloth typically has more folds and wrinkles. Intricate detail is also visible on a Greek sculpture (right) sculpted to resemble a light fabric.

[1]. Geometric collisions (using swept primitives) can resolve complex interactions accurately, but they become intractable as the resolution increases. Repulsion-based approaches (even with untangling) are not robust enough as the approximate geometric information they use degrades as simplex density increases. Even robust hybrid approaches such as [10] still require the use of a geometric collision algorithm for every time integration step that contains a collision, which becomes intractable as the mesh resolution increases and the time steps become smaller. We propose an extension to the hybrid repulsion/collision technique, defining a history-based repulsion/attraction scheme that allows us to rely less on geometric self collisions while still remaining fully robust, allowing tractable scaling to higher resolutions. Notably, the knowledge of a collision free state enables the application of smarter repulsion/attraction forces without heuristics to estimate the untangled configuration (e.g. voting algorithms and methods that preclude the use of edge/edge collisions).

The third contribution we make is introducing more accurate friction handling between cloth and collision objects to ensure that the extra simulated resolution is used effectively. Cloth object friction and interaction is especially important as the object a cloth is interacting with defines much of its behavior. We propose a novel technique for cloth-object collision and friction that is significantly more accurate than previous methods applied to a semi-implicit or implicit time stepping scheme. See Figure 10.

2 Previous Work

Computer graphics cloth simulation extends back at least 20 years, and early examples include [11], [24]–[29]. A good background on cloth modeling is provided in [30]. We use a variant of the semi-implicit method introduced by [2], but other examples of time integration for cloth include [12], [31]–[35]. Our goal is to obtain folds and wrinkles in a physically based fashion from the interplay of in-plane constitutive forces and bending forces, as opposed to adding the wrinkles via a separate modeling system, e.g. [3], [36], [37]. Although we do not address constitutive models for in-plane forces, our preliminary tests show that finite elements and mass-springs models behave similarly, but we refer the interested reader to [38], [39]. Bending models have been addressed by [2], [40]–[44]. Although we currently use a static resolution grid, an adaptive approach would allow even more resolution, see e.g. [45]. For collision detection, we use straightforward algorithms and extensions based on well known work but refer the interested reader to [7], [8] and the references therein. We also note the work on improving efficiency in low curvature regions in [22], [46].

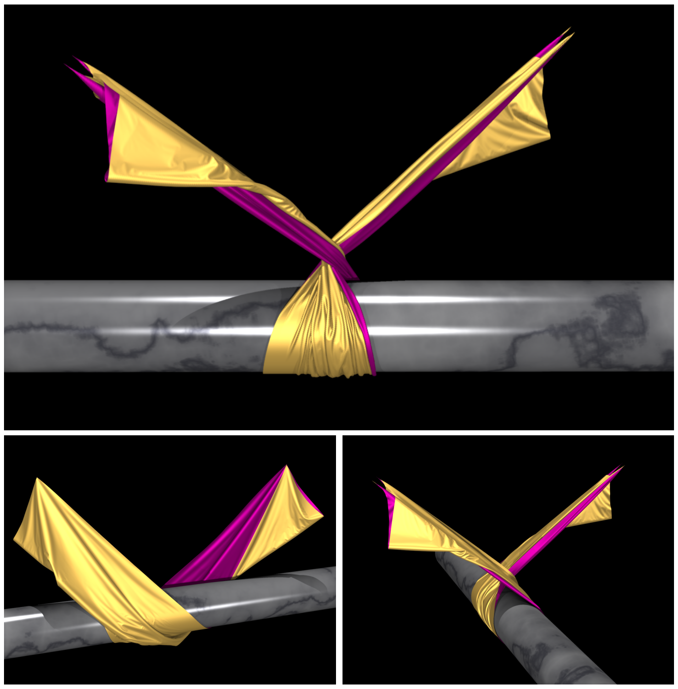


Fig. 3. A piece of cloth with a half-million triangles is forced to twist by a rotating cylinder. Even under such high tension, the cloth remains self-intersection free showing the robustness of our algorithm at high resolution.

3 Algorithm Overview

We use a simple mass-spring constitutive model with edge springs as well as bending springs that connect unshared vertices of adjacent triangles (illustrated in Figure 4) so we can simulate arbitrary triangle meshes. The force for a spring is

$$\boldsymbol{F} = \left[E\left(\frac{\|\boldsymbol{u}(t)\|}{\|\boldsymbol{u}(0)\|} - 1\right) + d(\boldsymbol{V_1}(t) - \boldsymbol{V_2}(t))^T \frac{\boldsymbol{u}(t)}{\|\boldsymbol{u}(t)\|}\right] \frac{\boldsymbol{u}(t)}{\|\boldsymbol{u}(t)\|}$$

where $\boldsymbol{u}(t) = \boldsymbol{X_2}(t) - \boldsymbol{X_1}(t)$, E is the Young's modulus (stiffness) and d is the damping parameter. More accurate models such as finite-element constitutive models

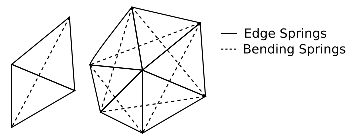


Fig. 4. Constitutive model of cloth consisting of both springs on triangle edges and springs across shared edge of triangle pairs.

are possible, but we are interested in accurately modeling collisions and interactions so we found this simple model to be sufficient. Although any integration scheme can be used, we use a semi-implicit Newmark variant similar to [2] which allows us to resolve the higher frequency elastic forces explicitly while efficiently handling the damping forces implicitly. The main differences are where we have placed self-repulsion and body collision updates.

Our approach to integrating time integration and collision detection/response is similar to [10], but with some key differences. As they did, we have an outer collision loop that puts the mesh into a collision free state. Within this outer collision loop a nested time integration scheme is used to produce a candidate final position and velocity for each particle in the mesh. The collision loop then calculates an effective velocity by subtracting the candidate position from the last collision free state and modifies this effective velocity until it obtains a displacement that removes all collisions. If collisions were found, [10] rewound to the last collision free state and repeated the simulation with half the number of time steps per collision step, eventually resulting in one time step per collision step. In Section 4 and Section 5, we describe how our history-based attraction/repulsion framework allows us to circumvent this difficulty and thus achieves better performance while maintaining robustness (see Figure 14). The ith iteration of our *outer collision loop* step proceeds as:

A. Compute repulsion pairs and their orientation history
B. Perform k_i time integrations (*inner loop*)
   1. $\tilde{v}^{n+1/2} = v^n + \frac{\Delta t}{2} a(t^{n+1/2}, x^n, \tilde{v}^{n+1/2})$
   2. Modify $\tilde{v}^{n+1/2}$ with elastic and inelastic self-repulsion
   3. $\tilde{x}^{n+1} = x^n + \Delta t \tilde{v}^{n+1/2}$
   4. Collide with body objects to obtain x^{n+1} and $v^n_\star$
   5. $v^{n+1/2} = v^n_\star + \frac{\Delta t}{2} a(t^{n+1/2}, x^{n+1/2}, v^{n+1/2})$
   6. Extrapolate $\tilde{v}^{n+1} = 2v^{n+1/2} - v^n_\star$
   7. Modify $\tilde{v}^{n+1}$ to v^{n+1} for friction with objects
   8. Modify v^{n+1} with friction and inelastic self-repulsion
C. Detect and resolve moving collisions.

Here Δt is the time step, $a(t, x, v)$ is the acceleration and $x^{n+1/2} = (x^n + x^{n+1})/2$. In step A we search hierarchies for repulsion pairs as described in Section 5. In step B we perform k_i time integrations which consists of several steps: step B.1 is a backward Euler solve to obtain a temporary velocity, which is subsequently modified with self-repulsions in step B.2 (Section 5), before being used to advance the cloth positions forward in time in step B.3. Then in step B.4 our novel cloth-object collision algorithm (Section 6) is applied, obtaining the final collision corrected position x^{n+1} and intermediate velocity $v^n_\star$. Next, we apply a backward Euler solve in step B.5 followed by an extrapolation in step B.6, which are equivalent to applying the trapezoidal rule to velocity but are significantly better conditioned than the standard formulation (see [47]). Finally, in step B.7 we modify this final velocity to obtain the appropriate cloth-object friction as dictated by our new cloth-object collision algorithm (Section 6) and subsequently apply self-repulsions in step B.8 (Section 5). Finally step C ensures that no collisions remain by detecting and removing collisions described further in Section 4.

We also employ distributed memory parallelism using message passing so that our algorithm can be used on more than one machine rather than being constrained to a single shared memory machine. Work is distributed across m processors by partitioning the particles into m disjoint sets. Our parallelization strategy and contributions are discussed in Section 7.

4 The Outer Collision Loop

During the course of a simulation, an outer loop of collision steps is performed where the ith collision step evolves time using k_i time integrations and then resolves collisions. As in [10] this loop maintains the invariant that positions are collision free at the beginning and end of each of these collision iterations. If $k_i > 1$ and a geometric collision was detected, [10] rewound time and reran the collision iteration with $k_i/2$ and this process would continue until $k_i = 1$. Only then were robust geometric self collisions used to zero the relative velocity of interacting collision pairs. We instead never rewind, and run with a fixed k_i between 8 and 16 ensuring tractability in complicated high-resolution interactions. [10], by contrast, applied repulsions only at the end of the collision step, using the stored self collision free positions. They could not apply repulsions per time step because as positions moved they had no way of tracking side coherence so repulsions could exacerbate tangling that produced more rigid groups and thus visual artifacts. The key to allowing per time step repulsions without creating artifacts is to store history information together with repulsions discovered in step A so that the repulsions applied per time in step B.2 and B.8 do the right thing (described in Section 5).

In step C, geometric collisions test whether an interaction pair (point/triangle or edge/edge) intersects by checking if the linear trajectories from the last known collision free positions to the current time integrated positions (after k_i time steps) interfere. The linear trajectory implicitly defines the notion of an effective velocity which equals the net change in position divided by the total time elapsed since the last collision free state. Potentially colliding pairs are found by box hierarchy searches with bounding boxes containing the swept trajectory primitives. Pairs of leaf boxes are further pruned by checking for intersections in a coordinate frame that moves with the average effective velocity of the involved particles.

Potentially colliding pairs are processed in Gauss-Seidel fashion using the algorithm in [10] which requires the solution of a cubic equation to determine if the four points involved become coplanar. The inelastically applied collisions are handled by zeroing the relative effective velocity and using this to compute the new positions. If the final positions of the collision pair are in too close proximity, an elastic self repulsion is applied (exactly as in self-repulsion) to push them further apart. We stress that, unlike in [10], this elastic repulsion impulse uses the average time step size taken during the collision loop as opposed to the total time elapsed during the collision loop because we can rely on future per time step repulsions. For any affected particle, we use its new position to calculate a new effective velocity.

After processing all potentially colliding pairs and obtaining new effective velocities, new collisions could be generated. Thus, we iterate the entire algorithm until no collisions are found. The second and later iterations are significantly less expensive than the first since we can reduce the cost of a hierarchy search by considering only box pairs whose expansion contains nodes modified in the previous iteration. As in [10], we rely on rigid "impact zones" when collisions cannot be resolved after a number of iterations. Not noted by other authors we found that rigid groups sometimes form between coplanar, colinear, or colocated points causing the inversion of the inertia tensor to fail. These situations occurred as we scaled to higher resolutions which explains why previous authors did not observe them. Thus we robustly compute the pseudoinverse of the symmetric 3×3 inertia tensor using a singular value decomposition. Once the algorithm finds no further collisions, step C is complete and we satisfy the collision-free invariant with the same robustness and visual quality as [10].

5 History-Based Self-Repulsions

Since geometric self collisions are expensive to compute and we would like to perform them less frequently, we rely on self-repulsions to help prevent and simplify resolution of collisions. Similar to collisions we consider both point/triangle and edge/edge interactions. Pairs are obtained in step A using a bounding box hierarchy which is discussed more extensively in Section 7. For any type of interaction, we first apply it for all point/triangle pairs followed by all edge/edge pairs, since there are fewer point/triangle pairs and their behavior is typically more robust. In step B.2 of the time integration loop, first an *inelastic collision* impulse is used to stop approaching interaction pairs, and then if necessary a spring-based *elastic repulsion* is used to push interaction pairs further apart. Note that the velocity used in step B.3 is discarded and that steps B.5 and B.6 fully integrate the velocity from time n to time $n+1$. Frictional effects are only calculated in step B.8 in order to implement self repulsions. The amount of friction is determined based on the normal forces caused by the inelastic repulsions used to stop cloth from encroaching on itself. Since elastic collisions can produce artifacts, we use the elastic repulsions only to modify the velocity that will be used to update the positions to reduce the chance of collisions while only using inelastic repulsions for the actual update to the velocity in step B.8.

[10] also made use of repulsions to ease the requirements on the geometric collision stage, but he applied them immediately before the collision stage (i.e. in our step C) using the linearized effective velocities. We instead use actual simulation velocity state and apply repulsions at per time step granularity resulting in increased stability and robustness. We follow [10]'s formulation of repulsions. Here we consider point-face, but edge-edge is similar. For an inelastic repulsion the impulse is $I_c = m\boldsymbol{v}_N/2$ where $\boldsymbol{v}_N$ is the normal velocity and m is the mass of all the pair's particles. For an elastic repulsion of spring stiffness k we use

$$I_r = -\min\left(\Delta t k d, m\left(\frac{0.1d}{\Delta t} - \boldsymbol{v}_N\right)\right) \quad (1)$$

where

$$d = h - (x_4 - w_1\boldsymbol{x_1} - w_2\boldsymbol{x_2} - w_3\boldsymbol{x_3})\cdot\hat{n} \quad (2)$$

and w_i are the barycentric weights of the free point x_4 projected to the triangle, $x_{\{1,2,3\}}$ are the triangle's point locations, h is the repulsion thickness, and $\hat{n}$ is the triangle normal. Note the elastic repulsion is limited to 0.1 of the interpenetration, removing the need for any damping parameter and the inelastic repulsion does not require damping as it provides infinite damping in the normal direction. The modification of the friction is also accomplished with an impulse that uses the change in normal velocity applied by the inelastic or elastic collision Δv_N.

Unfortunately, repulsions only work if the interaction pairs contain the correct notion of sidedness, as they otherwise work against an interference free state if the pairs have crossed (e.g. if a point spuriously crosses a face). Several authors have suggested switching to attractions (e.g. [1]) after cloth has non-physically interpenetrated itself, however it can be difficult to ascertain whether attractions or repulsions should be applied and thus sometimes the interactions are turned off altogether. Since crossing may occur during the time step, checking for interpenetration at discrete times is insufficient. [10] avoids this problem by only applying repulsions in the collision free state, which has the downside of only allowing repulsions at the granularity of the outer collision loop. Since mesh elements can move considerably during the time integration, many potential repulsions are missed, reducing the benefit of repulsions in avoiding collisions as well as reducing the amount of small scale bending and folding that could be produced.

Our key idea is to compute and store interaction data from the collision free state in step A and to subsequently

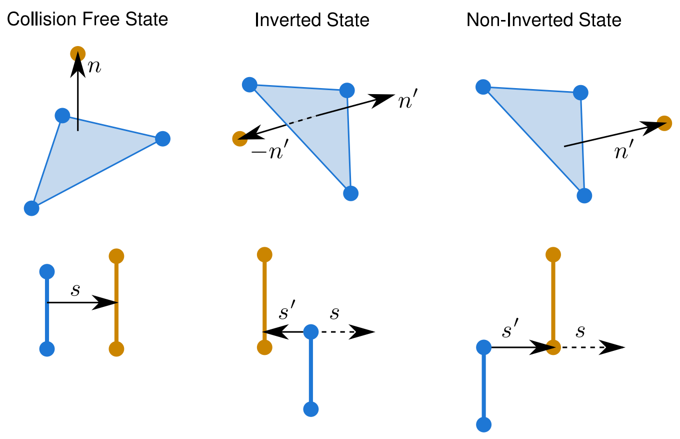


Fig. 5. Inversion criteria for point/triangle and edge/edge interaction pairs. (Top) Point/triangle pairs are considered inverted if the point is in a different triangle half-plane. (Bottom) Edge/edge pairs are considered inverted if the shortest direction vector rotates too far from its collision free orientation.

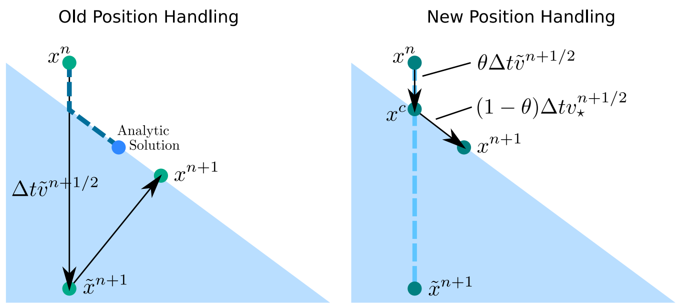


Fig. 6. Particles are depicted falling towards an incline plane. (Left) A simple level set adjustment to position incorrectly projects in the normal direction. (Right) Our improved method first finds the collision point and then steps to the final position.

use this data to apply history-based repulsions and attractions during all time integration steps during step B. This increases efficiency because we do not apply a per time step collision detection scheme but instead detect all potentially interacting repulsion pairs during the outer collision loop. Although this tends to produce more pairs than is necessary, and in fact we purposefully use non-aggressive pruning to capture more pairs because the elements can move significantly during a sequence of time steps, it is quite efficient to perform a simple prune at each time step to see if the potentially interacting pairs are indeed interacting. The total cost of all operations in our per time step repulsion method (including detection and application) is typically 7% of the code run time. This approach contrasts with others' who have used voting schemes to construct sidedness (e.g. [48]) as our use of the collision free state to determine orientation alleviates the need for majority (voting) approaches.

During the outer collision loop in step A, after finding all potentially interacting pairs, we compute and store the relative orientation for later use in our history-based repulsion scheme. For the point/triangle case, we apply a repulsion whenever the point remains on the correct side of the triangle as determined by its normal (Figure 5 top), and otherwise switch to an attraction. This is implemented by ensuring the normal $\hat{n}$ always points toward the free point at the collision free time. If not then the points are reordered. i.e. if we have pair (x_1, x_2, x_3, x_4) then if $(x_2 - x_1 \times x_3 - x_1) \cdot (x_4 - x_1) < 0$ then consider the pair as (x_1, x_2, x_4, x_3). Then as a point x_4 passes from the correct side through the triangle, d will increase in equation 2, leading to a larger impulse in a correcting direction in equation 1. For the edge/edge case, we store the shortest direction vector between the two interacting edges in the collision free state $\boldsymbol{s_0}$, and later compare this with the shortest direction vector $\boldsymbol{s}$ in the current state (Figure 5 bottom). This formulation does not penalize rotation of segments in parallel planes, but it does penalize rigid body rotations. This makes the edge/edge history-based heuristic less reliable than the point-face one so we only flip the current shortest vector if $\boldsymbol{s_0} \cdot \boldsymbol{s} < \epsilon$ (where $\epsilon = 0$ is aggressive and $\epsilon = .9$ is conservative).

6 Cloth-Object Collisions

Bridson [2] suggested a level set based collision algorithm that processes each position $\tilde{x}^{n+1}$ and velocity v^n as follows. Suppose $\phi(x)$ measures the signed distance from a point in space to all objects in the scene $\phi(x) < 0$ is inside and $\phi(x) > 0$ outside. Then if $\phi(\tilde{x}^{n+1}) < 0$, a collision is detected with depth $d = |\phi(\tilde{x}^{n+1})|$ and normal $N = \nabla\phi(\tilde{x}^{n+1})$, assuming that $\nabla\phi$ has already been normalized. The position is then projected in the normal direction $x^{n+1} = \tilde{x}^{n+1} + dN$ as shown in Figure 6 (left). For the sake of exposition the following steps define a function $v_\star = \Gamma(v)$ that adjusts the velocity to remove any inward normal component and include the effects of friction. The normal and tangential velocity components are computed as $v_N = v^T N$ and $v_T = v - v_N N$ for the particle and $v_{BN} = v_B^T N$ and $v_{BT} = v_B - v_{BN} N$ for the body. Here v_B is the velocity of the body at time $n+1$. The modified normal velocity $v_{\star N} = \max(v_N, v_{BN})$ ensures that the relative velocity does not point into the body. The tangential velocity is modified to $v_{\star T} = v_{BT} + \max\left(0, 1 - \mu\frac{v_{\star N} - v_N}{|v_{T,rel}|}\right) v_{T,rel}$ where $v_{T,rel} = v_T - v_{BT}$ and μ is the coefficient of friction. The final result is

$$v_\star = v_{\star N} N + v_{\star T}.$$

The two sources of inaccuracy in this previous method emanate from errors in the position update obtained by projecting in the normal direction and the subsequent changes in velocity incurred during the conjugate gradient solve. Although [2] constrained the normal velocity during their second conjugate gradient solve, the friction and motion in the tangential direction were still adversely affected. While infinite friction could be obtained by also constraining the tangential velocity, one cannot accurately obtain finite nonzero friction, and moreover one cannot constrain the velocity in the first conjugate gradient solve as it would cause cloth to stick to objects.

To compute a more accurate collision adjusted position, we find the point where the level set changes sign $x^c = x^n + \theta\Delta t\tilde{v}^{n+1/2}$ where $\theta = \phi(x^n)/(\phi(x^n) - \phi(\tilde{x}^{n+1}))$ by linear interpolation. If the object is moving, ϕ depends on time as well, and we replace $\phi(x^n)$ with $\phi(x^n) + \Delta t v_{BN}$ in the definition of θ noting that all evaluations of the level set function ϕ occur with the time $n+1$ collision body. Next we compute $v_\star^{n+1/2} = \Gamma(\tilde{v}^{n+1/2})$ which is used to move from x^c to the final modified position $x^{n+1} = x^c + (1-\theta)\Delta t v_\star^{n+1/2}$. Note that $(1-\theta)\Delta t$ is the remaining fraction of our time step after the collision as shown in Figure 6 (right). Although $v_\star^{n+1/2}$ yields the correct post collision velocity for use in the position update, we also compute $v_\star^n = \Gamma(v^n)$ for subsequent use in the trapezoidal rule (steps B.5 and B.6) which is more correct at time n. While steps B.1 to B.3 of the time integration scheme are used to obtain the correct position, steps B.5 and B.6 are used to update the velocity which also must be corrected for contact. Specifically, any forces applied during the trapezoidal rule velocity update will contain both tangential and normal components so that the conjugate gradient algorithm must project the normal components of the forces to zero to keep points in contact constrained to $v_{\star N}^n$.

Similar to [12] we can determine the net normal force applied during the trapezoidal rule (or for backward Euler) and use it to correct our friction algorithm. Although [12] used a simple model, we use the more complex Γ function. For the sake of exposition, assume that the final velocity step was a backward Euler step to t^{n+1} instead of the trapezoidal rule, then the velocity update equation is

$$\tilde{v}_P^{n+1} = v_\star^n + P(\frac{\Delta t}{m}F_i(t^{n+1}, x^{n+1}) + \frac{\Delta t}{m}F_d(t^{n+1}, x^{n+1})\tilde{v}_P^{n+1})$$

where F_i is the velocity independent force, $F_d\tilde{v}^{n+1}$ is the linear velocity dependent damping force, P projects away the normal component of each colliding point with $I - NN^T$, and $\tilde{v}_P^{n+1}$ are the solved velocities (which have normal components constrained by projections). Using the same forces without applying the projection gives a different result

$$\tilde{v}_{NP}^{n+1} = v_\star^n + \frac{\Delta t}{m}F_i(t^{n+1}, x^{n+1}) + \frac{\Delta t}{m}F_d(t^{n+1}, x^{n+1})\tilde{v}_P^{n+1}$$

which includes the global effects of the projection in F_d but applies all forces to the velocities without projection. Although $\tilde{v}_{NP}^{n+1}$ does not have a zero normal relative velocity for nodes in contact, its damping forces have been properly globally conditioned for contact, i.e. they use the $\tilde{v}_P^{n+1}$ velocity. To include contact in step B.7, we take $\tilde{v}_{NP}^{n+1}$ and apply Γ to obtain $v^{n+1} = \Gamma(\tilde{v}_{NP}^{n+1})$. Note that Γ will compute the same normal force that P applied during the conjugate gradient solve. In addition, Γ will apply the appropriate friction that P did not apply.

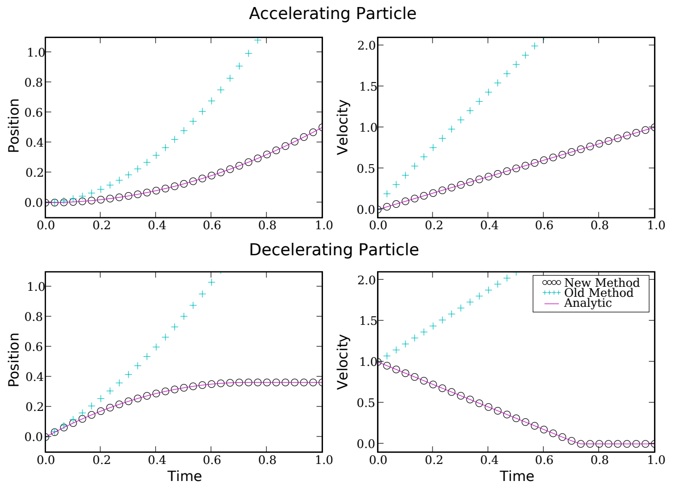


Fig. 7. Plot of displacement and tangential velocity over time for a single particle moving down an incline plane. Our result is coincident with the analytic result for both accelerating and decelerating particles, while the previous method accelerates too fast in both cases. In particular the analytic velocity is $v(t) = v_0 + (\sin\theta - \mu\cos\theta)gt$ when not stopped. If $\mu > \tan\theta$ stopping happens when $v(v_0/(g(\sin\theta - \mu\cos\theta))) = 0$. The analytic position is $x(t) = v_0 t + (\sin\theta - \mu\cos\theta)gt^2/2$. We use $\theta = \pi/5$, $g = 9.8$. For accelerating we use $v_0 = 0$ and $\mu = .6$ and for decelerating we use $v_0 = 1$ and $\mu = .9$

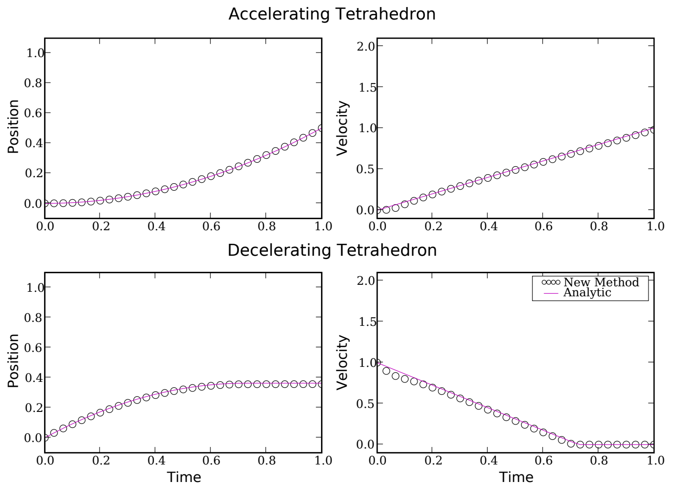


Fig. 8. A single tetrahedron slides down an incline plane, matching the analytic solution for both accelerating and decelerating test cases. The analytic formulas and parameters are the same as the particle's.

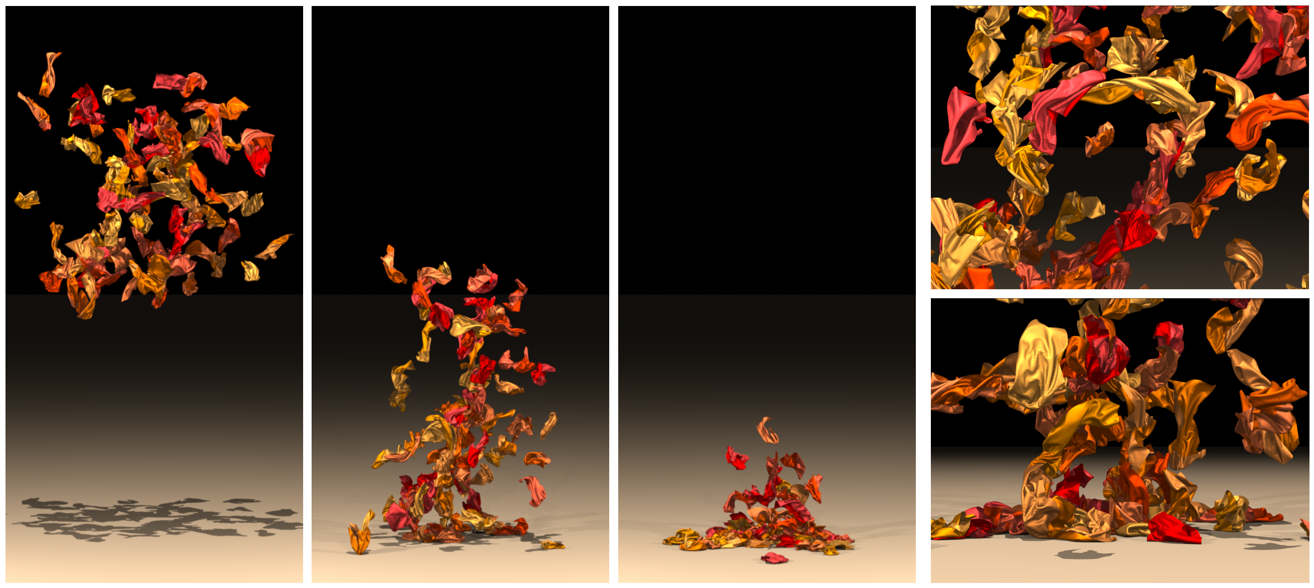


Fig. 9. We drop 100 pieces of cloth with 10,000 triangles each (1 million triangles total). A linear wind drag model causes them to flutter and interact. The simulation time was under 5 minutes per frame.

To test our new algorithm we consider a single particle sliding down an incline plane. We consider two cases: one with a particle slowing down and coming to rest and another with a particle starting from rest and accelerating. (Note in these simple cases, the incline plane does not move, the normal does not change and the particle starts and stays in contact, i.e. $\theta = 0$) Figure 7 compares the previous algorithm, our improved version, and the analytic solution. Since the single particle test case does not require conjugate gradient, as gravity is the only force, we show in Figure 8 the same test repeated with a tetrahedron that uses non-trivial damping forces. Figure 10 shows a more complicated example where cloth rolls with static friction.

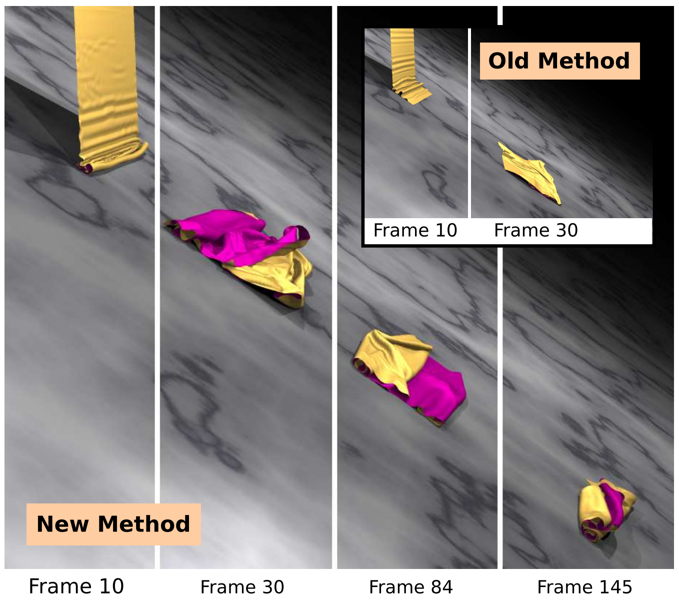


Fig. 10. A piece of cloth with initial tangential velocity falls on an incline plane with high coefficient of friction, and undergoes static friction causing it to roll. The cloth simulated by the previous method slips immediately on contact with the ground and moves faster (incorrectly) down the incline. Note the resolution here is about 80,000 triangles, showing that even on lower resolutions we achieve better results.

If many collision objects are used during a simulation, the cost of evaluating ϕ can be prohibitive, especially for memory-intensive collision objects which we desire to process only once. For efficiency we use a uniform spatial partition for collision body occupancy and iterate over cloth points, creating a list of potential interactions. Subsequently, each collision body is accessed only once and all potentially interacting points are processed with it.

One limitation of this algorithm is that the linearizations used for the position correction can be problematic on high curvature objects. Another limitation of this algorithm is that it queries for point penetration within the collision body, so if a collision body is excessively thin or velocities are high, a collision might be missed. In practice this is rarely an issue and in fact this type of collision approach is frequently used for a character's body in order to simulate clothing. Alternatively, body collisions can be handled within our self-collision framework instead, although the much more efficient cloth-body algorithm should be favored if it is applicable.

7 Parallelism

Each particle in the cloth mesh is assigned to a processor by using a recursive median split. If the cloth configuration changes significantly, it may improve performance

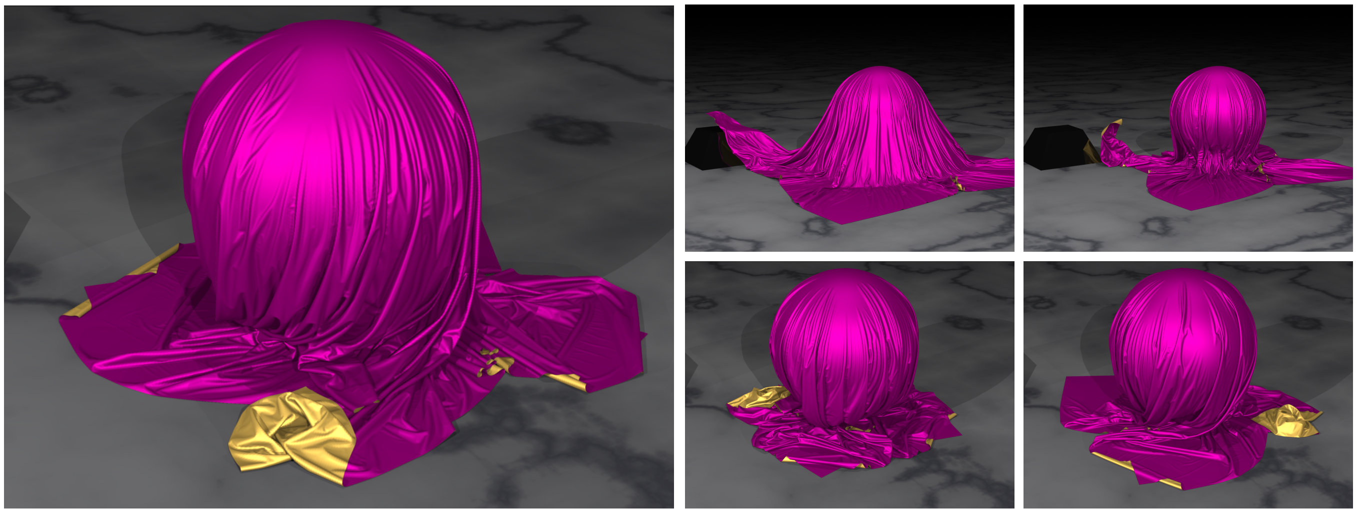


Fig. 11. A piece of cloth with 1.8 million triangles is draped over a ball. Low ground friction causes the cloth to tightly pack beneath the sphere. As the ball begins to spin, the high frictional coefficient causes the detailed folds of the cloth to twist.

to reassign particles using the median split again and transfer particle data accordingly; we found this unnecessary in our examples. Forces involving particles across processor boundaries are computed redundantly on each processor owning involved particles after first sending all boundary particle information across processors. It is significantly cheaper to exchange the state and compute forces redundantly than to compute the forces on one processor and send the results to other processors. Conjugate gradient also requires inter-processor communication to perform inner product and residual norm reductions.

In steps A and B of the outer loop we perform searches for pairs of interacting objects using axis-aligned bounding box hierarchies [49]. We synchronize the position data to every processor so each can construct a full hierarchy for points, segments, and triangles, which we use for doubly recursive traversals on every processor. Point/triangle interaction pairs are obtained by colliding the point hierarchy with the triangle hierarchy and edge/edge pairs are obtained by colliding the segment hierarchy against itself. We assign the detection of each potentially interacting pair to the processor of lowest index that owns one of the involved particles. Each processor searches for its assigned interaction pairs by pruning pairs of boxes whose expansion only contains interaction pairs assigned to other processors. Thus we avoid searching down branches of the Cartesian product tree if any interactions found would be owned by other processors.

Applying repulsions or collision responses must also be parallelized but it is important to make sure pair response is done in Gauss-Seidel ordering. That is, each pair should be processed seeing the newest data that is available, so the effect of any previously applied repulsion is used in any subsequent repulsion with which it shares nodes. If response is done instead in Gauss-Jacobi order then impulses may be counted multiple times. While others [14] have have discussed parallelization of cloth, these papers have not discussed the importance of collision response ordering. Thus our parallel algorithm proceeds in two passes whenever we wish to apply a collision or repulsion set. We label interaction pairs (point/triangle or edge/edge) involving particles owned by different processors as *boundary pairs* and label those owned by only one processor as *internal pairs*. Using a flood-fill algorithm, each connected component of boundary pairs is processed separately (in parallel) in the first phase noting that a Gauss-Seidel ordering is still used within each connected component. Next in the second phase, modified particle velocities are sent to the processors that own the respective particles, and the remaining internal pairs are processed (again in Gauss-Seidel order) independently by the processor that contains them. See Figure 12 for an illustration of this application strategy. Note that this strategy always ensures the effective parallel ordering is equivalent to some serial Gauss-Seidel ordering. We note that repulsion pairs, boundary pairs and internal pairs are discovered and transferred to the appropriate processors in step A.1 and are used many times in step B.2 and B.8.

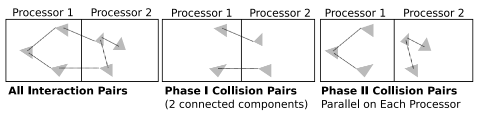


Fig. 12. Parallel collision/repulsion pair Gauss-Seidel ordering. Phase I consists of each connected component of boundary pairs being processed (middle). Phase II computes all internal pairs using the new data from the boundary pair processing.

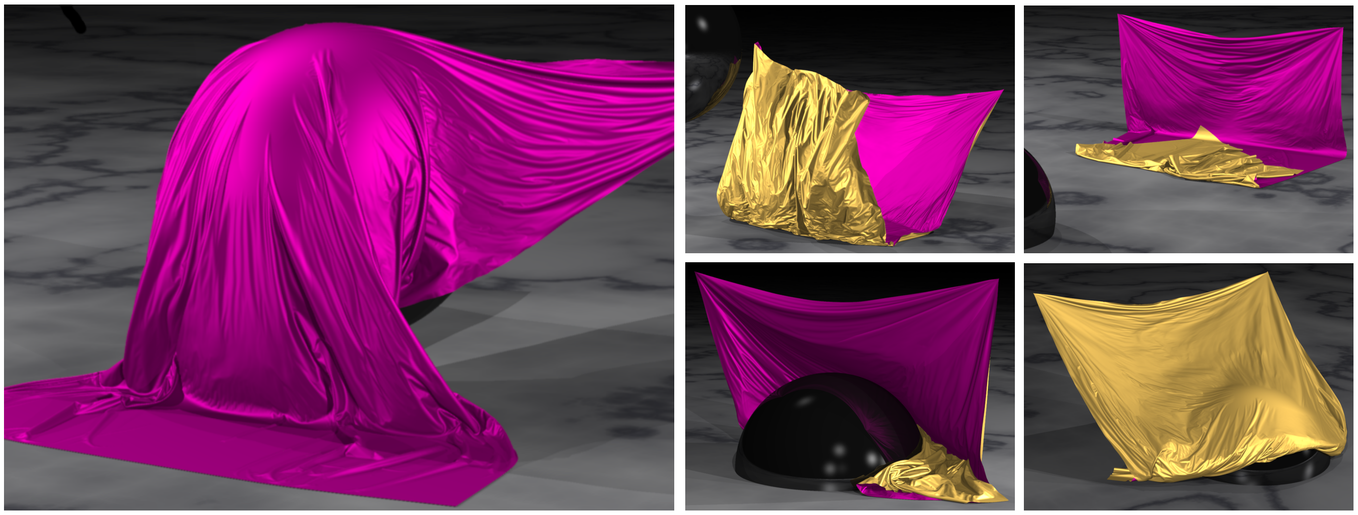


Fig. 13. A piece of cloth with 1.7 million triangles is flipped by a sphere causing it to fold over itself. The sphere then pushes through the complicated clump of folds.

8 Examples

We ran our simulations with meshes ranging from a half-million to 2 million triangles on between 2 to 4 quad-processor Opteron 2.8 GHz machines connected by gigabit ethernet, but obtained similar performance profiles on commodity dual-core dual-processor machines. We used 16 time steps between each collision processing step though we reduced this number to 4 or 8 during a few collision intensive sections. Automatically choosing the number of time steps between collision processing steps is important future work but we emphasize that approaches that scale to one time step per collision loop whenever collisions occur [10] or use one time step per collision loop [21] become intractable on large meshes. Even on lower resolution meshes where there are fewer collisions, we still get a benefit by running with 16 total loops (e.g. Figure 14). None of our examples used viscous air (ether) drag to damp the velocities, and only the simulations in Figures 9 and 15 used wind drag. We rendered cloth using a standard ray tracer. Although the results of cloth simulations can have their resolution artificially increased in a subdivision postprocess (even avoiding collisions as in [10]), our simulated examples were of sufficient resolution to require no subdivision except for the example that used 100 pieces of lower resolution 10 thousand triangle cloth. This is an important step as subdivision is only a stop-gap measure that makes renderings visibly smooth, and it cannot introduce detail missing from the low-resolution simulation. We have summarized some of the key parameters that we used to generate our examples in Table 1.

We begin with our lowest resolution example, a half-million triangle version of a twisting cloth torture test shown in Figure 3, demonstrating the robustness of our algorithm even in difficult collision situations. This example averaged 30 minutes per frame, but as the two ends become intertwined the strain on the knotted self-collision area becomes very high and frames take

Parameter	Description	Value
E_e	Edge Youngs Modulus (Stiffness)	$4.14\ \mathrm{N}$
ξ_e	Edge Overdamping Fraction	15
E_b	Bending Youngs Modulus (Stiffness)	$0.41\ \mathrm{N}$
ξ_b	Bending Overdamping Fraction	8
CG Tol	Convergence for Backward Euler	$10^{-3}\ \mathrm{m/s}$
Mass	Mass of node in mesh	$10^{-6}\ \mathrm{kg}$
g	Gravitational constant	$9.8\ \mathrm{m/s^2}$
h	Repulsion Thickness	$0.01\ \mathrm{m}$
k	Repulsion Spring Constant	$30\ \mathrm{Ns/m}$
k_i	Collision Total Loops	$4 - 16$

TABLE 1
Parameters of our model.

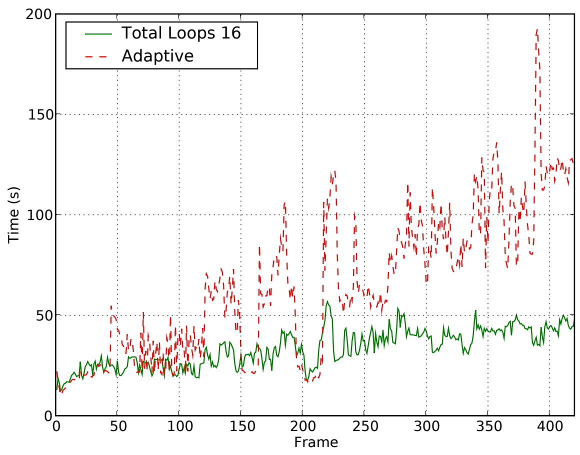


Fig. 14. Timing comparison of using adaptive collision resolution with rewind (Bridson) and the fixed total loop strategy that we can apply due to our history based repulsion scheme.

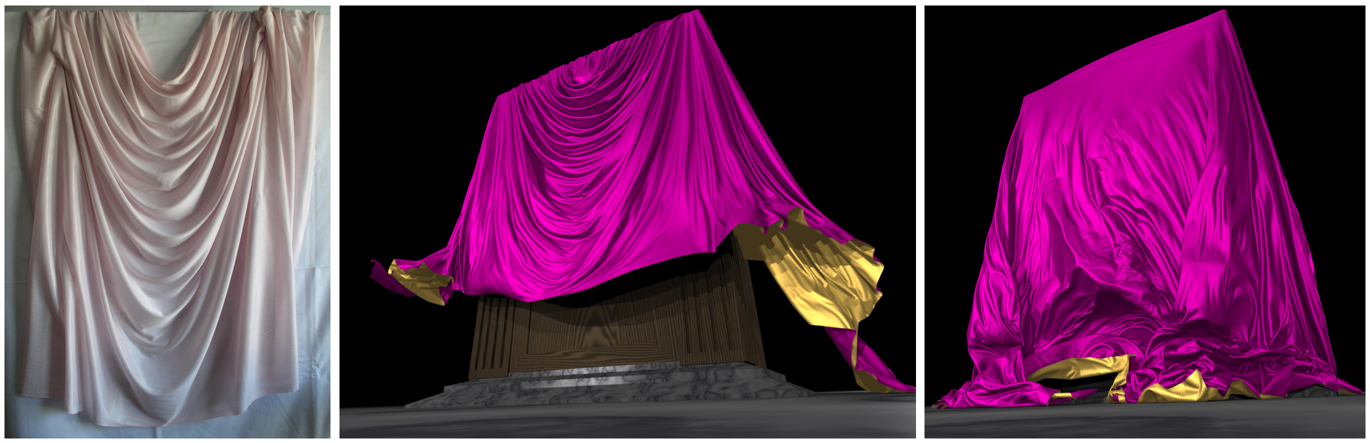


Fig. 15. (Left) A photograph of a real piece of cloth draped showing intricate folds and wrinkles. The middle and right depicts a synthetic piece of cloth with 2 million triangles draped over a wardrobe. The coefficient of friction between the cloth and the wardrobe is reduced, causing the cloth to fall. Small scale wrinkles and fluttering are introduced by our wind drag model as shown in the right image. Notice how the synthetic example has more folds and wrinkles which is due to it representing a larger size piece of cloth.

longer to complete. In Figure 9 we demonstrate that we can handle many pieces of cloth, simulating 100 separate falling cloths with 10 thousand triangles each, totaling 1 million triangles. We employ a wind-drag model that produces interesting deformation and flutter, facilitating inter and intra-cloth interactions. For each vertex, we apply a linear drag in the normal direction as a simple approximation to the pressure force, noting that a better approximation would be quadratic in velocity. We maintain a high wind speed near the ground in order to push the lightweight pieces of cloth around, causing further interactions and increasing the number of dynamic collisions that must be resolved even after the pieces of cloth hit the ground. At 5 minutes per frame across two machines, this was our lowest cost simulation.

The next examples demonstrate a single high resolution mesh with simulation parameters chosen to accentuate folds and wrinkles and their subsequent collisions and interactions. Figure 11 shows the spinning ball example from [8], [10] with an increased resolution of 1.8 million triangles. To promote a very high level of detail we avoid using overly stiff cloth and air drag, use high sphere but relatively low ground friction, and raise the vertical position of the sphere slightly so that the initial draping of the cloth results in even more self collisions at the base of the sphere. This example averaged 20 minutes per frame. Figure 13 shows the curtain and ball example from [10] at a resolution of 1.7 million triangles. The multiple layers of contact that form when the cloth folds over itself make this example particularly difficult, especially when the sphere pushes through the clump of layers. The average frame time for the simulation was 45 minutes. Figure 15 shows a 2 million triangle cloth draped over a wardrobe. We initially animated several vertices of the cloth to obtain a more interesting draping effect. After an initial settling period, we change the coefficient of friction on the wardrobe allowing the cloth to fall and form many folds and wrinkles under the effect of wind drag. The cloth in this simulation resembles the photograph of draped cloth in Figure 15 and the satin depicted in Figure 2. This simulation averaged just over 6 minutes per frame. A summary of the resolutions and timings of our examples is shown in Table 3.

We found parallelism essential as it allowed us to scale to very high resolutions although it alone was not responsible for our results. In fact we first parallelized the [10] algorithm but found it insufficient, motivating our other improvements. An example of speedup (on the curtain and ball example) is shown in Figure 16. A break-

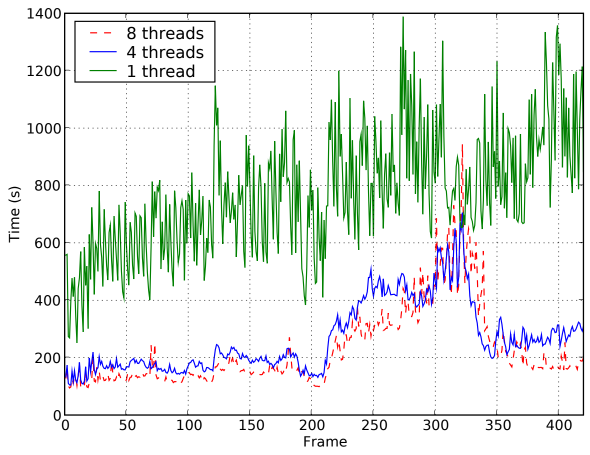


Fig. 16. Comparison of performance of curtain and ball simulation (Figure 13) with increasing numbers of processors. We also ran this simulation with 16 threads, though there was not a significant speed up over 8 threads. This is most likely due to our slow gigabit ethernet interconnect.

Simulation Step	% of total sim	% of subpart
Compute repulsion pairs/history (step A)	3-7%	
Inner time integration loop (step B)	45-60%	
Applying per time step repulsions		2-4%
Detect & resolve collisions (step C)	37-48%	
Applying collisions		20-45%
Computing swept bounding boxes		5-20%
Traversing hierarchies		5-15%
Inter-processor communication		40-70%

TABLE 2
A breakdown of the time spent in our algorithm.

Example	Resolution	Avg. frame time	Processors
Twisting Cloth	0.5 million	30 mins	16
Cloth Leaves	1 million	5 mins	8
Spinning Ball	1.8 million	20 mins	16
Curtain & Ball	1.7 million	45 mins	16
Wardrobe	2 million	6 mins	16

TABLE 3
List of our examples with their resolutions, average time per frame, and number of processors they were run on.

down of where time is spent during our simulations is shown in Table 2. Roughly speaking, half of the CPU time is dedicated to collisions, while half of that (about 25% of the total CPU time) is spent on inter-processor communication. This is probably due to our slow gigabit ethernet interconnect. As commodity hardware vendors continue the trend of adding more cores to a processor with dedicated internal interconnects, the situation should improve.

9 Conclusion

We have addressed several issues in cloth simulation to allow simulating high resolution cloth, enabling us to represent and simulate intricate folds and wrinkles. Using a multi-processor approach on commodity hardware we demonstrated simulations of cloth with up to 2 million triangles. To make collisions more efficient on these large meshes, we employ a history-based attraction/repulsion scheme that takes advantage of the last known collision free state. Applying the repulsions/attractions at every time step reduces the frequency at which the more expensive geometric collision algorithm is used. In addition, to ensure resolution is used effectively, we correctly handle cloth-object friction which facilitates formation and preservation of folds and wrinkles at both low and high resolutions.

Acknowledgment

Research supported in part by a Packard Foundation Fellowship, an Okawa Foundation Research Grant, ONR N00014-01-1-0620, ONR N0014-06-1-0393, ONR N00014-06-1-0505, ONR N00014-02-1-0720, ONR N00014-05-1-0479 for a computing cluster, ARO DAAD19-03-1-0331, NIH U54-GM072970, NSF ACI-0323866, NSF IIS-0326388, NSF ITR-0205671 and NSF CCF-0541148.

References

[1] D. Baraff, A. Witkin, and M. Kass, "Untangling cloth," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 22, pp. 862–870, 2003.
[2] R. Bridson, S. Marino, and R. Fedkiw, "Simulation of clothing with folds and wrinkles," in *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2003, pp. 28–36.
[3] L. Cutler, R. Gershbein, X. Wang, C. Curtis, E. Curtis, C. Curtis, E. Maigret, and L. Prasso, "An art-directed wrinkle system for CG character clothing," in *Proc. ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2005.
[4] F. Cordier, H. Seo, and N. Magnenat-Thalmann, "Made-to-measure technologies for an online clothing store," *IEEE Comput. Graph. and Appl.*, vol. 23, no. 1, pp. 38–48, Jan 2003.
[5] P. Decaudin, D. Julius, J. Wither, L. Boissieux, A. Sheffer, and M.-P. Cani, "Virtual garments: A fully geometric approach for clothing design," in *Comp. Graph. Forum (Eurographics Proc.)*, 2006.
[6] R. McDonnell, S. Dobbyn, S. Collins, and C. O'Sullivan, "Perceptual evaluation of LOD clothing for virtual humans," in *Proc. ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2006.
[7] N. Govindaraju, D. Knott, N. Jain, I. Kabul, R. Tamstorf, R. Gayle, M. Lin, and D. Manocha, "Interactive collision detection between deformable models using chromatic decomposition," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 24, no. 3, pp. 991–999, 2005.
[8] A. Sud, N. Govindaraju, R. Gayle, I. Kabul, and D. Manocha, "Fast proximity computation among deformable models using discrete Voronoi diagrams," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 25, no. 3, pp. 1144–1153, 2006.
[9] P. Volino and N. Magnenat-Thalmann, "Resolving surface collisions through intersection contour minimization," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 25, no. 3, pp. 1154–1159, 2006.
[10] R. Bridson, R. Fedkiw, and J. Anderson, "Robust treatment of collisions, contact and friction for cloth animation," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 594–603, 2002.
[11] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically deformable models," *Comput. Graph. (Proc. SIGGRAPH 87)*, vol. 21, no. 4, pp. 205–214, 1987.
[12] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *ACM SIGGRAPH 98*. ACM Press/ACM SIGGRAPH, 1998, pp. 43–54.
[13] M. Keckeisen and W. Blochinger, "Parallel implicit integration for cloth animations on distributed memory architectures," *EG Symposium on Parallel Graphics and Visualization*, 2004.
[14] B. Thomaszewski and W. Blochinger, "Parallel simulation of cloth on distributed memory architectures," *EG Symposium on Parallel Graphics and Visualization*, 2006.
[15] B. Thomaszewski, S. Pabst, and W. Blochinger, "Exploiting parallelism in physically-based simulations on multi-core processor architectures," *EG Symposium on Parallel Graphics and Visualization*, 2007.
[16] R. Lario, C. Garcia, M. Prieto, and F. Tirado, "Rapid parallelization of a multilevel cloth simulator using openmp," *In Third European Workshop on OpenMP*, 2001.
[17] S. Romero, L. Romero, and E. Zapata, "Rapid parallelization of a multilevel cloth simulator using openmp," *EuroPar*, 2000.
[18] F. Zara, F. Faure, and J.-M. Vincent, "Physical cloth animation on a pc cluster," *In Fourth Eurographics Workshop on Parallel Graphics and Visualisation*, 2002.
[19] X. Provot, "Collision and self-collision handling in cloth model dedicated to design garment," *Graph. Interface*, pp. 177–89, 1997.
[20] S. Huh, D. N. Metaxas, and N. I. Badler, "Collision resolutions in cloth simulation," in *Comput. Anim.* IEEE, 2001.
[21] S. Huh and D. Metaxas, "A collision resolution algorithm for clump-free fast moving cloth," in *Proc. of Comp. Graph. Intl.*, 2005.
[22] P. Volino, M. Courchesne, and N. Magnenat-Thalmann, "Versatile and efficient techniques for simulating cloth and other deformable objects," *Comput. Graph. (SIGGRAPH Proc.)*, pp. 137–144, 1995.
[23] P. Volino and N. Magnenat-Thalmann, "Accurate collision response on polygonal meshes," in *Proc. of Comput. Anim.*, 2000, pp. 154–163.
[24] J. Weil, "The synthesis of cloth objects," *Comput. Graph. (SIGGRAPH Proc.)*, pp. 49–54, 1986.

[25] D. Terzopoulos and K. Fleischer, "Modeling inelastic deformation: viscoelasticity, plasticity, fracture," *Comput. Graph. (SIGGRAPH Proc.)*, pp. 269–278, 1988.

[26] J. A. Thingvold and E. Cohen, "Physical modeling with B-spline surfaces for interactive design and animation," *Comput. Graph. (SIGGRAPH Proc.)*, pp. 129–137, 1992.

[27] M. Carignan, Y. Yang, N. Magnenat-Thalmann, and D. Thalmann, "Dressing animated synthetic actors with complex deformable clothes," *Comput. Graph. (SIGGRAPH Proc.)*, pp. 99–104, 1992.

[28] H. Okabe, H. Imaoka, T. Tomiha, and H. Niwaya, "Three dimensional apparel CAD system," *Comput. Graph. (SIGGRAPH Proc.)*, pp. 105–110, 1992.

[29] D. E. Breen, D. H. House, and M. J. Wozny, "Predicting the drape of woven cloth using interacting particles," *Comput. Graph. (SIGGRAPH Proc.)*, pp. 365–372, 1994.

[30] D. H. House and D. E. Breen, Eds., *Cloth modeling and animation*. A. K. Peters, 2000.

[31] M. Meyer, G. Debunne, M. Desbrun, and A. H. Barr, "Interactive animation of cloth-like objects in virtual reality," *The Journal of Visualization and Computer Animation*, vol. 12, no. 1, pp. 1–12, 2001.

[32] D. Parks and D. Forsyth, "Improved integration for cloth simulation," in *Proc. of Eurographics*, ser. Comput. Graph. Forum. Eurographics Assoc., 2002.

[33] E. Boxerman and U. Ascher, "Decomposing cloth," in *Proc. ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2004, pp. 153–161.

[34] P. Volino and N. Magnenat-Thalmann, "Implicit midpoint integration and adaptive damping for efficient cloth simulation," *Computer Animation and Virtual Worlds*, vol. 16, pp. 163–175, 2005.

[35] S. Oh, J. Ahn, and K. Wohn, "Low damped cloth simulation," *Visual Computer*, vol. 22, no. 2, Feb. 2006.

[36] S. Hadap, E. Bangarter, P. Volino, and N. Magnenat-Thalmann, "Animating wrinkles on clothes," *Proc. of Visualization*, pp. 175–523, 1999.

[37] M.-H. Choi, M. Hong, and S. Welch, "Modeling and simulation of sharp creases," in *SIGGRAPH 2004 Sketches & Applications*. ACM Press, 2004.

[38] O. Etzmuss, M. Keckeisen, and W. Strasser, "A fast finite element solution for cloth modelling," in *Pacific Graph.*, 2003, pp. 244–251.

[39] G. Irving, J. Teran, and R. Fedkiw, "Invertible finite elements for robust simulation of large deformation," in *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2004, pp. 131–140.

[40] K.-J. Choi and H.-S. Ko, "Stable but responsive cloth," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 21, pp. 604–611, 2002.

[41] E. Grinspun, A. Hirani, M. Desbrun, and P. Schröder, "Discrete shells," in *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2003, pp. 62–67.

[42] M. Bergou, M. Wardetzky, D. Harmon, D. Zorin, and E. Grinspun, "A quadratic bending model for inextensible surfaces," in *Proc. of Eurographics Symp. on Geometry Processing*, 2006, pp. 227–230.

[43] B. Thomaszewski, M. Wacker, and W. Strasser, "A consistent bending model for cloth simulation with corotational subdivision finite elements," in *Proc. ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2006.

[44] P. Volino and N. Magnenat-Thalmann, "Simple linear bending stiffness in particle systems," in *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2006, pp. 101–105.

[45] E. Grinspun, P. Krysl, and P. Schröder, "CHARMS: A simple framework for adaptive simulation," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 21, pp. 281–290, 2002.

[46] P. Volino and N. Magnenat-Thalmann, "Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity," in *Proc. of Eurographics*, ser. Comput. Graph. Forum, vol. 13. Eurographics Assoc., 1994, pp. C–155–166.

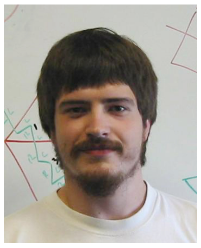
[47] E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw, "Hybrid simulation of deformable solids," in *Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2007, pp. 81–90.

[48] P. Volino and N. Magnenat Thalmann, "Collision and self-collision detection: Efficient and robust solutions for highly deformable surfaces," in *Comp. Anim. and Simulation*, D. Terzopoulos and D. Thalmann, Eds. Springer-Verlag, 1995, pp. 55–65.

[49] G. van den Bergen, "Efficient collision detection of complex deformable models using AABB trees," *J. Graph. Tools*, vol. 2, no. 4, pp. 1–14, 1997.

Andrew Selle received his B.S. in Computer Science and Mathematics from the University of Wisconsin in 2003. While there, he was an undergraduate research assistant working on computer animation research. He plans to receive his Ph.D. in Computer Science at Stanford University in 2008. While there, he also was a consultant at Intel Corporation, focusing on physical simulation's mapping to multicore architectures. He has also been a consultant for Industrial Light + Magic for the last two years, receiving screen credits for his work on "Poseidon," "Evan Almighty" and "Pirates of the Caribbean: At World's End." He will begin work at Disney Animation in Summer, 2008.

Jonathan Su received his B.S. in Computer Engineering from the University of Washington in 2006. He was awarded the National Science Foundation Graduate Research Fellowship and is currently pursuing his Ph.D. at Stanford University. For the past year he has been a consultant at Intel Corporation where he has been studying the scalability of physical simulation algorithms on multicore architectures.



Geoffrey Irving received his Ph.D. in Computer Science from Stanford University in 2007, and his B.S. in Mathematics and Computer Science from the California Institute of Technology in 2003. He was awarded a National Science Foundation Graduate Research Fellowship and three Caltech Upperclass Merit Awards. For the last three years at Stanford, he was a consultant at Pixar Animation Studios focusing on fluid dynamics and elastic character simulation, and received screen credits for his work on "Ratatouille" and the upcoming film "WALL•E." He is currently a member of the Production Engineering Group at Pixar.

Ron Fedkiw received his Ph.D. in Mathematics from UCLA in 1996 and did postdoctoral studies both at UCLA in Mathematics and at Caltech in Aeronautics before joining the Stanford Computer Science Department. He was awarded an Academy Award from The Academy of Motion Picture Arts and Sciences, the National Academy of Science Award for Initiatives in Research, a Packard Foundation Fellowship, a Presidential Early Career Award for Scientists and Engineers (PECASE), a Sloan Research Fellowship, the ACM Siggraph Significant New Researcher Award, an Office of Naval Research Young Investigator Program Award (ONR YIP), the Okawa Foundation Research Grant, the Robert Bosch Faculty Scholarship, the Robert N. Noyce Family Faculty Scholarship, two distinguished teaching awards, etc. Currently he is on the editorial board of the Journal of Computational Physics, Journal of Scientific Computing, SIAM Journal on Imaging Sciences, and Communications in Mathematical Sciences, and he participates in the reviewing process of a number of journals and funding agencies. He has published over 80 research papers in computational physics, computer graphics and vision, as well as a book on level set methods. For the past seven years, he has been a consultant with Industrial Light + Magic. He received screen credits for his work on "Terminator 3: Rise of the Machines", "Star Wars: Episode III - Revenge of the Sith", "Poseidon" and "Evan Almighty."