Efficient Yarn-based Cloth with Adaptive Contact Linearization

Jonathan M. Kaldor Doug L. James Steve Marschner
Cornell University

Figure 1: Character-Scale Garments: *A character wearing a knit sweater containing 45,960 yarn loops walks forward. By exploiting spatial and temporal coherence in the contact forces, we focus computation on rapidly deforming parts of the model (model updates in red).*

Abstract

Yarn-based cloth simulation can improve visual quality but at high computational costs due to the reliance on numerous persistent yarn-yarn contacts to generate material behavior. Finding so many contacts in densely interlinked geometry is a pathological case for traditional collision detection, and the sheer number of contact interactions makes contact processing the simulation bottleneck. In this paper, we propose a method for approximating penalty-based contact forces in yarn-yarn collisions by computing the exact contact response at one time step, then using a rotated linear force model to approximate forces in nearby deformed configurations. Because contacts internal to the cloth exhibit good temporal coherence, sufficient accuracy can be obtained with infrequent updates to the approximation, which are done adaptively in space and time. Furthermore, by tracking contact models we reduce the time to detect new contacts. The end result is a 7- to 9-fold speedup in contact processing and a 4- to 5-fold overall speedup, enabling simulation of character-scale garments.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: knitted, cloth, yarn, contact, adaptive, corotational

1 Introduction

Sheet-based cloth simulation is ubiquitous in animation, visual effects, and apparel design. Yarn-based simulations promise major quality improvements for many cloth types, but they are more expensive to simulate for two reasons. First, sheet models can readily be coarsened for faster simulation, but yarn models always need enough detail to describe the shape of the yarn, so reducing the number of degrees of freedom is not straightforward. Second, sheet models only generate contact processing work when the sheet collides with objects or folds over and collides with itself, but yarn models derive their whole behavior from the thousands of self-collisions within the fabric's structure. In fact, Kaldor et al. [2008] report spending the majority of simulation time in contact processing. In this paper, we present a method for accelerating yarn-based simulation by significantly reducing the cost of contact processing.

Simulating yarn contacts accurately is critical because the structure of the cloth is formed entirely by the interlacing or interlooping of the yarns—if contacts are missed and yarns are allowed to pass through each other, the cloth will unravel.[1] In many ways, this contact-mediated structure of rods in close proximity is a worst-case scenario for collision processing. Prior work has resolved these compressible yarn-yarn contacts using stiff penalty forces, requiring small timesteps with an all-pairs evaluation over the entire yarn at each step [Kaldor et al. 2008].

Fortunately, some key characteristics of yarn models also make contact processing easier. Internal contacts are coherent: they tend to persist throughout the simulation, and the local yarn shape changes slowly (in its local frame of reference), with individual contacts exhibiting near-rigid motion. Consequently, the contact force, although stiff, is temporally coherent, suggesting reuse of contact information over the course of many simulation steps.

[1] Compare this to hair, where even if collisions are missed and hairs pass through each other, the resultant state is still a valid system configuration.

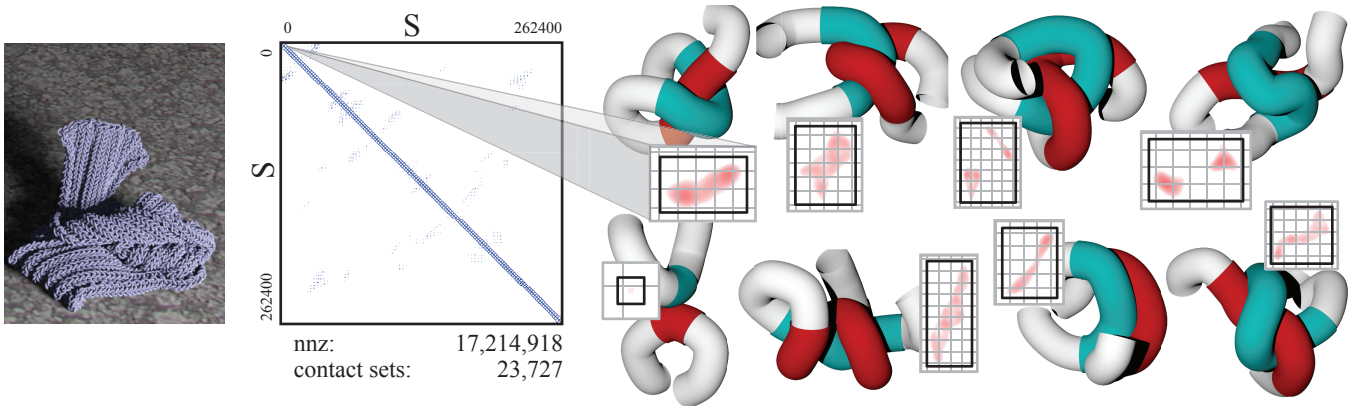


Figure 2: Contact Zoo: *Although the space for potential contacts is huge, only a very small fraction of it is actually in contact, as seen in the contact matrix. Due to the looping nature of knits, though, these contacts can take on a variety of unique shapes, shown with the actual pairs of points in contact in the contact matrix (in red) and the corresponding contact set (outlined in black). All of these contacts were taken from the single timestep of the falling scarf shown on the left.*

In this paper, we propose a corotational force approximation to the yarn's penalty-based contact force. The exact contact response is computed for a particular configuration, and then a rotated linear force model is used to approximate the force under small deformations (see Figure 3). Once the shape changes too much, a new model is built centered at the new configuration. The contact response for the cloth is broken into many sub-contacts (see Figure 2), each involving two contiguous sections of yarn.

Temporal adaptivity is controlled by a single quality parameter that determines how frequently the contact models are rebuilt—setting it to zero causes exact force evaluations at every step, resulting in zero approximation error. A space-time scheduler efficiently finds new contacting regions while potentially managing tens of millions of spline segment pairs. Finally, by explicitly tracking contact features, we also reduce self-collision detection costs for existing close-proximity yarn segments.

2 Prior Work

Cloth simulation can be divided into two broad categories: sheet-based models and yarn-based models. Sheet-based models ignore the yarn-based structure of cloth and approximate the overall behavior as an elastic sheet [Terzopoulos et al. 1987; Baraff and Witkin 1998; Choi and Ko 2002]. These simulators are typically significantly faster than yarn-based simulators and tend to work well for woven fabrics. Their speed and reliability has improved through enforcing inextensibility [Provot 1995; Goldenthal et al. 2007; English and Bridson 2008], improved bending models [Bridson et al. 2003; Grinspun et al. 2003], simplified bending models that exploit isometric sheet deformations [Bergou et al. 2006; Garg et al. 2007], or more robust collision processing [Volino and Thalmann 2000; Bridson et al. 2002; Baraff et al. 2003], such that interactive simulations are possible for models of modest complexity.

In contrast, yarn-based simulators use yarns as their primary simulation primitive and rely on yarn-yarn contact forces to mediate deformations. These have included models and simulators focused on the yarn crossings in woven fabrics [Kawabata et al. 1973], modeling plain-weave fabrics like Kevlar [King et al. 2005; Zeng et al. 2006], and simulating different types of woven [Chu 2005] and knit fabrics [Kaldor et al. 2008]. These simulators automatically capture nonlinear deformations that are difficult to duplicate in sheet based simulators, particularly knits, at the cost of a significant increase in computation time. They rely on an underlying model for thin flexible rods, which have been studied extensively in computer graphics [Pai 2002; Bertails et al. 2006; Theetten et al. 2007; Bergou et al. 2008; Bergou et al. 2010]. Contacts between rods can be resolved precisely using inequality constraints [Spillmann and Teschner 2008], but this does not allow for lateral compression of soft yarns as penalty-based models do [Kaldor et al. 2008], requiring explicit modeling via additional degrees of freedom.

Our rotated linear model for penalty-based contact forces is related to corotational finite-element methods commonly used in graphics for solid deformation. Müller et al. [2002] popularized these techniques in graphics, initially via node-based "stiffness warping," then, to overcome undesirable "ghost forces" due to element-level momentum imbalances, using rotated linear elements [Müller and Gross 2004] (c.f. [Felippa 2000]). Corotational elements have also been used for sheet-based cloth simulation [Etzmuss et al. 2003]. Shape-matching methods also use rotated frames to estimate "goal positions" for deformation forces [Müller et al. 2005; Rivers and James 2007]. Similar to Müller et al. [2002], our corotational force model is node based; however, it does not produce ghost forces since rotated contact sets are composed of contact pairs whose internal action-reaction forces can produce no net force.

Model reduction techniques have been devised to accelerate sheet-based cloth and other deformable models in graphics by effectively reducing the number of simulated degrees of freedom. Effective methods have been proposed for spatially adaptive mass-spring systems [Hutchinson et al. 1996], sheet-based cloth models [Villard and Borouchaki 2005], and rod simulations that resolve challenging contact configurations such as knot tying [Spillmann and Teschner 2008]; space-time adaptive simulation of deformable models can resolve localized contacts efficiently for real-time simulation [Debunne et al. 2001]; and multi-scale basis refinement can reduce meshing issues for spatial adaptation [Grinspun et al. 2002]. Unfortunately, multi-resolution/adaptive approximations are difficult for knitted cloth given its complex geometric domain and topology, high number of degrees of freedom, and widespread self-contact.

Homogenenization techniques have been proposed to coarsen discrete simulation models while resolving inhomogeneous material response [Kharevych et al. 2009], and to support deformation of complex embedded geometry [Nesme et al. 2009], but neither addresses fine-scale internal forces which are contact mediated. Dimensional model reduction techniques have been proposed to generate fast, low-rank simulation models for complex geometric domains [Barbič and James 2005; An et al. 2008] and thin shells [Chadwick et al. 2009], but knitted cloth motions do not necessarily lie in low-dimensional subspaces, and specifying them *a*

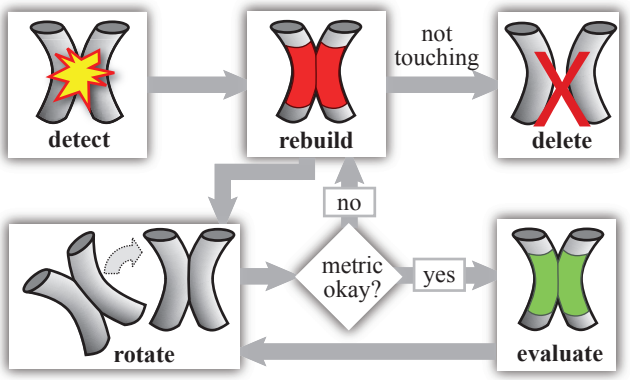


Figure 3: The Life of a Contact: *Once a new collision is detected, a linear contact model is built in the current frame. On subsequent timesteps, the model's control points are rotated into this reference frame. If the deformation is small according to our metric, the approximate force is evaluated, otherwise the model is rebuilt in its new configuration. If the contact separates too far, it is deleted.*

priori for precomputation purposes would be impractical. Recently, Kim and James [2009] showed how to adapt subspace deformation models on the fly to avoid precomputation; however, they do not address contact forces. In contrast to these model reduction techniques, we maintain the number of simulation degrees of freedom to preserve fundamentally interesting fine-scale yarn deformations.

Cloth self-collision detection (SCD) is commonly performed using overlap tests accelerated by spatial subdivisions, hash tables, or bounding volume hierarchies [Teschner et al. 2005]. Unfortunately many prior SCD acceleration schemes are inherently sheet-based, e.g., curvature tests [Volino et al. 1995], normal cones [Provot 1997], and chromatic decompositions [Govindaraju et al. 2005].

Identifying and tracking persistent contacts is related to space-time scheduling and collision processing [Mirtich 2000; Guibas 2004], which have been widely considered for maintaining proximity information for moving objects [Hubbard 1995; Gao et al. 2004]. Mirtich's Timewarp method [2000] investigated related strategies for collision detection and management of persistent contact groups for asychronous rigid-body simulation. In contrast, we use synchronized time-stepping to manage space-time collisions, but adaptively update contact groups and linearize contact-group force models. For deformable models, Harmon et al. [2009] used asynchronous contact mechanics and infinitely nested potentials to adaptively simulate penalty-based contact; however, their emphasis was on correctness for general scenarios, and not performance for hundreds of thousands of persistent yarn contacts.

3 Yarn-based Cloth Model

Our underlying yarn-based cloth model is an extension of Kaldor et al. [2008]. For simplicity we assume a single yarn, but extending support to multiple yarns is trivial. The centerline of the yarn is defined by a time-varying cubic Catmull-Rom spline $\boldsymbol{y}(s) = \sum_{i=0}^{n+1} B_i(s)\mathbf{q}_i$ for $s \in [0, n]$, and $\mathbf{q} = \mathbf{q}(t) \in \mathbb{R}^{3n+3}$ the control points of the cubic spline. The yarn is treated as inextensible with constant radius r, and each segment has a rest length ℓ_i.

3.1 Elastic Rods

Our contact model can be used with any rod model. In our system, we use Discrete Elastic Rods [Bergou et al. 2008] with a few modifications. Yarns are represented as inextensible rods with an isotropic bending response but a non-straight rest configuration. Inextensibility constraints and bending and twisting energies are computed using a piecewise linear discretization of the rod, while the contact response force is computed using the Catmull-Rom interpolating spline through the vertices of the linear discretization. The underlying contact force model is identical to Kaldor et al. [2008].

In Bergou et al. [2008], the material frames on each rod segment were described as a scalar rotation from a zero-twist frame, the Bishop frame, defined at each edge. For nonisotropic or naturally curved rods, this zero-twist frame inherently depends on the global state, and the derivative of the bending energy at any point in the yarn depends on the position of every prior point. Despite this dependence, the derivative of bending energy can still be computed in $O(n)$ time instead of $O(n^2)$. Unfortunately, we found that recursive bending energy computations were difficult to parallelize, and very long yarns (such as in garments) could produce large end-to-end rotations in the reference frame per timestep which can complicate endpoint orientation constraints.

We observe that twist-free reference frames were only used to simplify twist-energy computation—since then the only twisting is due to the material frame. Instead, our reference frame starts as a twist-free frame, but *every segment's frame* is parallel transported through time (instead of space) from its previous position. Because there is no more spatial parallel transport, rod energy computations have local support and are easily parallelized. Our reference frame does accumulate twist, but we can account for and correct our twisting energy computation by this additional twist (see Appendix A for details, or Bergou et al. [2010] for an alternate derivation).

Yarns tend to display significant plastic behavior under deformation. To approximate this, we use a simple plasticity model on the rest state of the rod in angular space. If the rest state (represented as a 2D point) at a segment/bending element pair lies outside the circle of radius p_{plastic} centered at the current state of that pair, the rest state is projected onto the boundary of the circle. Similarly, if the rest state falls outside the circle of radius $p^{\text{max}}_{\text{plastic}}$ centered at the origin, it is projected onto the boundary of the circle. This allows for permament deformations of the rest state within some allowable angular range at each bending element.

3.2 Penalty Contact Forces

We use as our reference contact model Kaldor et al. [2008], which used a penalty force derived from the contact energy,

$$E = \int_0^n \int_0^n \kappa(s, s')\, f(\boldsymbol{y}(s), \boldsymbol{y}(s'))\, ds\, ds',$$

$$f(\boldsymbol{y}, \boldsymbol{y}') = \begin{cases} \frac{4r^2}{\|\boldsymbol{y}-\boldsymbol{y}'\|_2^2} + \frac{\|\boldsymbol{y}-\boldsymbol{y}'\|_2^2}{4r^2} - 2, & \|\boldsymbol{y}-\boldsymbol{y}'\| < 2r \\ 0, & \text{otherwise} \end{cases}$$

where $\kappa(s, s') = \kappa_{\text{contact}}\, \ell_s \ell_{s'}$. In practice, this integral is discretized and solved via quadrature over all pairs of bn points, $\{s_i\}$:

$$E = \sum_{i=0}^{bn} \sum_{j=0}^{bn} \kappa_{ij}\, w_{ij}\, f(\boldsymbol{y}_i, \boldsymbol{y}_j)$$

where w_{ij} are the quadrature weights and b is the number of quadrature evaluations per yarn segment; $\boldsymbol{y}_i = \boldsymbol{y}(s_i)$, and $\kappa_{ij} = \kappa(s_i, s_j)$. Let $S = [0, \ldots, bn]$ be the set of quadrature points. Then this is a double sum over all quadrature points in the yarn, or a computation over the set of all elements in $S \times S$, which is exceedingly expensive to compute naïvely[2]. However, many of the points are not in contact ($f(\boldsymbol{y}_i, \boldsymbol{y}_j) = 0$) and can be safely skipped (see Figure 2).

[2]This computation is clearly symmetric since the work over (i, j) is the same as (j, i), so in practice we only compute the upper triangular part of this space. We use $S \times S$ here for notational simplicity.

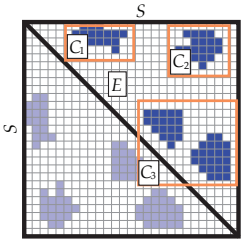
3.3 Yarn/Rod Timestepping

For further quality and speed improvements, our yarn simulator has several notable changes from prior work [Kaldor et al. 2008; Bergou et al. 2008]. Glue and length constraints are separated and solved individually using Fast Projection [Goldenthal et al. 2007]; because of the linear discretization, the linear systems are simple to solve directly (diagonal and tridiagonal, respectively). Only one phase of nonrigid damping is applied on each timestep, over blocks of 4×4 knit loops, and the damping is no longer always proportional to the nonrigid velocity. For a given block, let k be the number of control points in the block, and $\mathbf{v}_{\text{nonrigid}} \in \mathbb{R}^{3k}$ be the nonrigid velocity of that block. The applied change to the velocity is then $-\mathbf{v}_{\text{nonrigid}} \min(1, \max(h\mu_{\text{prop}}, h\mu_{\text{const}}\sqrt{k}/\|\mathbf{v}_{\text{nonrigid}}\|_2))$. At low speeds, rather than removing a proportional amount of nonrigid velocity at each step, a constant amount is removed, up to the entire nonrigid velocity; this allows the cloth to come to rest in finite time, which is an approximation of the complex hysteresis seen in real cloth. Finally, both damping and object contact are applied after glue and length constraints to allow the cloth to come to rest; while this means that the constraints are in principle no longer satisfied exactly at the end of each timestep, we have not noticed any problems in practice. See Figure 4 for details.

4 Adaptive Contact Linearization

We now describe the adaptive contact linearization force model used to avoid the penalty-based contact force computation (3.2) at each timestep. Our approximate contact model divides the collision region into a set of disjoint contacts and adaptively constructs simplified models for each contact. These simplified models are used as a cheap approximation of the true contact force for a set of configurations near some reference configuration. When the current contact configuration strays too far from the reference one, the contact model is discarded and a new one is constructed on the fly.

Contact Sets: Individual contacts are defined by partitioning $S \times S$ into disjoint sets $E, C_1, \ldots, C_m$, where E is empty space, and C_k is contact set k. We only require that every pair of points (i, j) that are in contact must be in a contact set: if $i, j \in S$ and $f(\boldsymbol{y}_i, \boldsymbol{y}_j) \neq 0$, then there is some contact k such that $(i, j) \in C_k$. Contact sets are allowed to be time-varying, and for the moment we simply assume that the sets are given; their construction and maintenance are addressed in §5.2. Given the contact sets, we can write the collision energy as a sum over contact features,



$$E(\mathbf{q}) = \sum_{k=0}^{m} E_k(\mathbf{q}) = \sum_{k=0}^{m} \left[\sum_{(i,j)\in C_k} \kappa_{ij} w_{ij} f(\boldsymbol{y}_i, \boldsymbol{y}_j) \right] \quad (1)$$

It follows that the total contact force is a sum over all contacts,

$$\mathbf{f}(\mathbf{q}) = -\nabla_{\mathbf{q}} E(\mathbf{q}) = \sum_{k=0}^{m} \mathbf{f}_k(\mathbf{q}), \quad (2)$$

with each contact's force a sum over contacting quadrature points,

$$\mathbf{f}_k(\mathbf{q}) = -\nabla_{\mathbf{q}} E_k(\mathbf{q}) = \sum_{(i,j)\in C_k} \kappa_{ij} w_{ij} \nabla_{\mathbf{q}} f(\boldsymbol{y}_i, \boldsymbol{y}_j). \quad (3)$$

Linearized Contact Forces: To avoid the expensive summation in (3), we can linearize any contact's force about a reference configuration, $\bar{\mathbf{q}}_k = \mathbf{q}(t_k)$, to obtain the approximation

$$\tilde{\mathbf{f}}_k(\mathbf{q}) = \mathbf{f}_k(\bar{\mathbf{q}}_k) + \bar{\mathbf{K}}_k (\mathbf{q} - \bar{\mathbf{q}}_k) \equiv \bar{\mathbf{f}}_k + \bar{\mathbf{K}}_k \mathbf{q}, \quad (4)$$

where $\bar{\mathbf{K}}_k$ is the stiffness matrix of $\mathbf{f}_k$ at $\bar{\mathbf{q}}_k$, and the force offset is $\bar{\mathbf{f}}_k = \mathbf{f}_k(\bar{\mathbf{q}}_k) - \bar{\mathbf{K}}_k \bar{\mathbf{q}}_k$. If contact C_k involves only c contacting quadrature points, then $\bar{\mathbf{K}}_k$ has at most $O(c^2)$ nonzero entries, which we compute and store in a dense triangular matrix format (exploiting both matrix and 3x3 block symmetry), along with $\bar{\mathbf{f}}_k$. Later times, $t > t_k$, can use (4) to quickly approximate $\mathbf{f}_k(\mathbf{q}(t))$.

Corotational Force Model: Since local contact geometry often undergoes large near-rigid deformations, we employ a corotational force approach analogous to [Müller and Gross 2004]. For each contact C_k, we estimate the nearest rigid transformation of the reference control points $\bar{\mathbf{q}}_k$ associated with C_k to the deformed configuration $\mathbf{q}$ by first matching the contact's center of mass and then finding its rotation R_k using the polar decomposition [Müller et al. 2005; Rivers and James 2007]. We then replace the linear force model $\tilde{\mathbf{f}}_k(\mathbf{q})$ in (4) by its corotational generalization,

$$\mathbf{f}_k(\mathbf{q}) \approx \mathbf{R}_k \tilde{\mathbf{f}}_k(\mathbf{R}_k^T \mathbf{q}) = \mathbf{R}_k \bar{\mathbf{f}}_k + \mathbf{R}_k \bar{\mathbf{K}}_k \mathbf{R}_k^T \mathbf{q}, \quad (5)$$

where $\mathbf{R}_k \in \mathbb{R}^{(3n+3)\times(3n+3)}$ is the matrix with the 3×3 matrix R_k repeated on the diagonal, and $\mathbf{q}$-translations omitted since $\bar{\mathbf{K}}_k$ annihilates them. Due to sparsity, only C_k-related control points and forces are evaluated in practice. Because R_k is updated on each timestep, we warm-start the Jacobi iteration used for the polar decomposition with the eigenvalues/vectors from the previous timestep [Rivers and James 2007]. By tracking near-rigid motion, the corotational contact force model (5) can be used longer than (4) before recomputation is necessary.

Model Invalidation: To avoid using (5) beyond its range of validity, we invalidate the contact model if it undergoes sufficient nonrigid deformation. If the current configuration, rigidly transformed back to the reference frame and denoted $\tilde{\mathbf{q}}_k$, has strayed too far from the reference configuration $\bar{\mathbf{q}}_k$, we rebuild the linear model about the current configuration, $\mathbf{q}$. The shape estimate, $\texttt{metric}(C_k)$, used to indicate contact invalidation is

$$\texttt{metric}(C_k) = \max_i M_{ik} 2r \|(\tilde{\mathbf{q}}_k)_i - (\bar{\mathbf{q}}_k)_i\|_2 / \epsilon_k^2 \quad (6)$$

where if $C_k = [u_{\min}, u_{\max}] \times [v_{\min}, v_{\max}]$, then M_{ik} is the weight for control point i in contact k, taken to be

$$M_{ik} = \int_{u_{\min}}^{u_{\max}} |B_i(s)| ds + \int_{v_{\min}}^{v_{\max}} |B_i(s)| ds,$$

and ϵ_k is the minimum distance between pairs of interacting quadrature points in C_k computed when the model was last rebuilt. Again, due to sparsity we only need to evaluate this metric for control points related to C_k. When this metric is larger than some user-supplied tolerance τ, we rebuild the model; we use $\tau = 0.004$–0.3. This metric is cheap to evaluate, allows greater movement for control points that weakly (or don't) influence contact C_k, and causes more frequent invalidations for close-proximity contacts.

5 Contact Adaptation

The contact algorithm is broken up into several phases, which can be broadly categorized into "contact detection" and "contact evaluation." The algorithm is summarized in Figure 4.

5.1 Contact Representation

From the description in §4, there is an obvious performance tradeoff in the construction of the contact sets C_k. Choosing to make smaller sets means that they are cheaper to update, but produces more sets to process, and increases the cost of set maintenance, for instance to determine when sets are overlapping.

$\mathbf{f}^{(t+1)} =$ `evaluate_other_forces()`
for each segment i
  `rasterize_to_grid(`i`)`
for each new segment/cell pair i, j
  `create_schedule(`i`, grid(`j`))`
for each schedule entry $sched$
  `process_schedule(`$sched$`)`
`coalesce_contacts()`
for each contact C_k
  if (`metric(`C_k`)` $> \tau$) `rebuild_model(`C_k`)`
  $\mathbf{f}^{(t+1)} = \mathbf{f}^{(t+1)} +$ `compute_force(`C_k`)`
$\dot{\mathbf{q}}_{\text{uncons}} = \dot{\mathbf{q}}^{(t)} + h\mathbf{M}^{-1}\mathbf{f}^{(t+1)}$
$\dot{\mathbf{q}}_{\text{glue}} =$ `satisfy_glue_constr(`$\mathbf{q}^{(t)} + h\dot{\mathbf{q}}_{\text{uncons}}$`)`
$\dot{\mathbf{q}}_{\text{length}} =$ `satisfy_length_constr(`$\mathbf{q}^{(t)} + h\dot{\mathbf{q}}_{\text{glue}}$`)`
$\dot{\mathbf{q}}_{\text{damp}} =$ `nonrigid_damping(`$\dot{\mathbf{q}}_{\text{length}}$`)`
$\dot{\mathbf{q}}^{(t+1)} =$ `object_contact(`$\mathbf{q}^{(t)} + h\dot{\mathbf{q}}_{\text{damp}}$`)`
$\mathbf{q}^{(t+1)} = \mathbf{q}^{(t)} + h\dot{\mathbf{q}}^{(t+1)}$
`quasistatic_frames()`

Figure 4: Algorithm Overview

Our implementation strikes a balance between these competing concerns. Contact sets are represented by bounding boxes in $S \times S$, and contain an estimate of a single contiguous contact between two parts of the yarn along with a specified amount of padding α to aid in detecting contact sliding (see Figure 5). Overlap tests are thus trivial to implement, and it provides a simple rule for when and how to merge contact sets. At the same time, in testing it seems to reasonably balance the number and size of contact sets while only including a relatively small number of non-contacting points.

5.2 Contact Detection

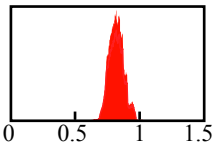
At each timestep, contact detection explores the empty contact set, E, to find newly colliding points and either (a) incorporates them into an existing contact if they overlap, or (b) creates a new contact. Given the sheer size of E, efficiency is of paramount importance. We do this through a combination of broad-phase spatial hashing of spline segments to find potential new contacts, and a space-time collision scheduler that can efficiently track millions of close-proximity spline segment-segment pairs.

Due to the sheer number of quadrature points and potential pairs, the scheduler tracks potential collisions between pairs of spline segments, each of which contains b quadrature points. However, our contacts exist at the quadrature point level, which means that between any two segments, some pairs of quadrature points may already belong to a contact set while others do not. Thus, in order to completely explore E while not spending time rediscovering already-known contacts, each schedule entry also stores an 8×8 bitmap, represented as a single long integer, which allows finer control over which pairs of quadrature points are already part of some existing contact and which still need to be checked. Finally, each schedule entry i stores a conservative minimum distance bound d_i^t which represents a best (but conservative) estimate for the closest possible distance between the two segments at time t.

Contact detection is divided into four phases: grid rasterization, newcomer scheduling, schedule processing, and contact coalescing

Grid Rasterization: The first step rasterizes each spline segment into a hashed uniform grid centered on the cloth's center of mass, with cell width γ. The AABB of each spline segment is computed in parallel by solving three quadratic equations to obtain its maximum and minimum extent in each of the three dimensions. Grid cells overlapping the AABB are marked as occupied by the segment. A segment not already belonging to a grid cell is flagged as a *newcomer* to the cell. At this time, the minimum and maximum movement and change in movement (the finite difference of movement) per timestep of the spline segment over the previous step are also computed (by solving six more quadratics) for later use by the scheduler. Note that if two segments occupy disjoint sets of grid cells, this serves as a certificate that the segments cannot be in contact. Since this is performed for all segments on every timestep, this serves to filter out any possible contacts between segments for which this is true.

On the right is a histogram of maximum AABB edge length for segments, normalized to the grid size of 0.6cm, for a single timestep of the sweater; nearly all segment bounding boxes are smaller than a single grid cell.



Newcomer Scheduling: In parallel, we check each cell newcomer against the cell's other segments to see if we already have a collision schedule entry for that newcomer-segment pair; if not, one is created and marked for immediate processing (by marking its minimum distance bound as negative). Collision schedule entries are deterministically assigned to one of the two participating segments for processing.

Space-Time Schedule Processing: Once all possible schedule entries are created, the schedules are looped over in parallel and executed. When an entry is determined to need full processing, the scheduler examines all active pairs of quadrature points on the two segments, computes the true minimum distance, and sets d_i^t accordingly.

For schedule entry i, the minimum distance estimate d_i^t between the pair of segments is used to determine when full processing is needed. As long as the minimum distance is positive, then they are known to not be in contact. The scheduler updates this minimum distance estimate and only processes the entry again when the estimate becomes negative. This minimum distance estimate is updated using the minimum and maximum movement computed during grid rasterization to determine the maximum relative movement $\Delta \boldsymbol{x}_i$ between the two segments over the previous timestep. One simple approach updates the minimum distance estimate each timestep according to the rule $d_i^t = d_i^{t-1} - \|\Delta \boldsymbol{x}_i\|_2$. This distance bound is guaranteed to always be less than or equal to the actual minimum distance, and so entries are guaranteed to be processed on time. Experimentally, this also succeeds in filtering out the vast majority of possible all-pairs checks per timestep, but it does incur the cost of examining and updating d_i^t for each schedule entry every timestep to determine if it needs to be processed further.

This cost of simply examining each schedule entry can begin to dominate the overall cost of schedule execution, however. To compensate for this, we divide the schedule into a set of bins, where each bin $\lambda \in [0, \lambda_{\max}]$ will now only be examined every 2^λ timesteps. Each segment now also stores the minimum and maximum movement $\|\Delta \boldsymbol{x}_i^\lambda\|$ over the previous 2^λ timesteps. When bin λ is examined, each schedule i in the bin updates its minimum distance bound $d_i^t = d_i^{t-2^\lambda} - \|\Delta \boldsymbol{x}_i^\lambda\|_2$, i.e., using the maximum relative movement over the time period in which it was not being examined. After either examining or processing a schedule, we assign it to a bin based on the current minimum distance, maximum relative movement $\Delta \boldsymbol{x}_i^\lambda$, and maximum relative change in movement $\Delta^2 \boldsymbol{x}_i$; we solve the quadratic equation

$$-d_i + \|\Delta \boldsymbol{x}_i^\lambda\| \frac{t}{h} + \frac{\left(\frac{t}{h}\right)^2 + \frac{t}{h}}{2} \left(\|\Delta^2 \boldsymbol{x}_i\| + \omega\right) = 0,$$

to find the minimum nonnegative root, then assign to bin $\max(0, \lfloor \log_2 \frac{t}{h} \rfloor)$. Because the relative velocity could change arbitrarily between checks, as in Hubbard [1995] we allow for some

bounded deviation in relative change in movement ω. When the relative change in movement between the two segments is larger than $\|\Delta^2 \boldsymbol{x}_i\| + \omega$, our predicted bin is no longer valid and we need to immediately update $d_i^{(t)}$ and reselect a new bin. However, we still wish to avoid having to examine every schedule pair each timestep. Instead, we monitor the local change in movement of each segment, and when it is more than $\frac{\omega}{2}$ from some reference change in movement, all schedules associated with that segment are immediately scheduled to have their minimum distance estimates updated, and the reference change in movement is updated to be the current change in movement. This conservatively reschedules contacts before their bin assignment becomes incorrect, while avoiding any loops over all schedules every timestep.

Contact Coalescing: The output of the schedule processor is a list of quadrature points $(i, j) \in S \times S$ that are currently in contact. The coalescer then takes these pairs (i, j) and groups them together into new contiguous contacts by floodfilling in $S \times S$ the axis aligned bounding box $(i - \alpha, j - \alpha)$ through $(i + \alpha, j + \alpha)$. Any overlapping contact sets (either new or pre-existing contacts) are merged into new AABBs until no more merges are required. This results in contact sets with a buffer of non-contacting pairs α around the detected contact, which allows the contact processor to detect when to resize a particular contact.

5.3 Contact Evaluation

After detecting new contacts, all contacts must be evaluated. This involves looping over the contact sets C_k in parallel, computing the best rigid transformation and evaluating the contact-invalidation metric. If the metric is not violated, we evaluate the current approximate model (5) via matrix-vector multiplication and vector addition. Because of the semi-regular looping structure of cloth, contiguous contacting regions tend to be small and localized, leading to small contact sets (which are designed to cover a single contact each); in practice, contact matrices $\mathbf{K}_k$ typically have between 24 and 63 rows/columns.

If the metric (6) is violated for contact k, we rebuild the linearized contact model $(\bar{\mathbf{f}}_k, \bar{\mathbf{K}}_k)$ for $\bar{\mathbf{q}}_k = \mathbf{q}$, which involves looping over all $(i, j) \in C_k$ to accumulate first and second derivatives of $f(\boldsymbol{y}_i, \boldsymbol{y}_j)$. Once this is done, corotational forces are evaluated using (5). While looping over the set, we also compute the minimum and maximum points $i_{\min}, i_{\max}, j_{\min}, j_{\max}$ that are in contact, and afterwards update C_k to cover $[i_{\min} - \alpha, i_{\max} + \alpha] \times [j_{\min} - \alpha, j_{\max} + \alpha]$ (see Figure 5). If there are no contacting points in the set we mark it as contributing zero force, and once the minimum inter-point distance becomes larger than some threshold (we use $2.1r$) then C_k is deleted and the pairs of points in C_k become part of the empty region, E.

We are also able to use our contact set structures to efficiently model additional yarn behaviors at negligible cost. As a simple example, we insert a small number (4) of stiction springs for each contact set to model the interactions due to entangled fibers. These springs are inserted at fixed locations within the contact set, and are broken and rebuilt once their energy exceeds a specified value.

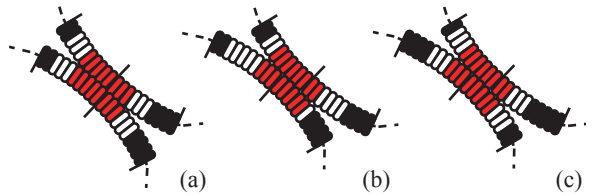
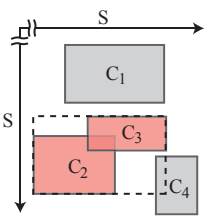


Figure 5: Contact Resizing *A contact (a) slides between updates (b), shifting the set of points in contact. A buffer allows us to detect this during update and readjust the contact to the new bounds (c).*

parameter	description	value
τ	approximation tolerance	0.004 – 0.3
α	flood-fill size	3
γ	grid size	0.6 cm
$\lambda_{\max}$	num. schedule bins	8
ω	movement change bound	0.0006 cm/timestep^2
h	timestep	1/16200s – 1/24000s
b	quadrature points / seg.	11
k_{contact}	contact stiffness	3000 – 4500
r	yarn radius	0.125 cm
p_{plastic} ; $p^{\max}_{\text{plastic}}$	yarn plasticity	0.01 ; 2.5
μ_{const} ; μ_{prop}	nonrigid damping	500 ; 1500 – 4500

Table 1: Parameters used during simulation.



Following this contact adaptation, we must determine whether adapted contacts are now overlapping. We divide S into a set of bins (in our implementation, each bin is of size 8) and place each C_k into each bin for which its first dimension overlaps. We then loop over each bin, scanning for overlaps only among sets in that bin, and merging any we find. For example, note that C_2 and C_3 are overlapping, which causes them to be merged together, which will then cause a cascading merge with the resultant merged contact and C_4. In principle, contacts should also be checked to determine if they have in fact split–that is, one contact set is now representing multiple contiguous contacts–since the cached dense stiffness matrix might become large with many zero entries. Because cloth has such regular structure, however, in practice we have not found the need to split contact sets into subpieces since contact splitting is a rare occurrence, and when it does happen the contacts stay reasonably close to each other (see Figure 2 for an example of a contact split), and so the matrices stay reasonable in size.

6 Results

Our simulator is in Java, multithreaded, and was run on a Mac Pro machine with two 4-core 2.93GHz Intel Xeon processors with 16GB of RAM. We compare against a reference implementation that computes the contact force via collision checks of a bounding hierarchy. In order to estimate scheduler performance, we also provide comparisons where the tolerance was zero (so models are rebuilt on every timestep) and stiffness matrix computation was disabled, providing a good approximation of the best possible speed obtainable when solving the contact forces exactly on each timestep. Timings and performance breakdowns are in Table 2 and Figure 6. Note that the average cost of contact coalescing was negligible ($<$ 1ms) in all simulations and is thus omitted from timings.

In order to verify our model, we compare the results of a scarf falling on a plane using both our reference implementation and our ACL model with a variety of tolerances (see Figure 9). Because for this example trivial deviations in force can result in drastically different behavior, variations in the final configuration are expected. For reasonable tolerances, we observe the simulations behave similarly and convey the same quality of motion. At higher tolerances the approximation becomes evident resulting in different material behavior, although it still is plausible. Note that the speedup we are able to achieve levels off, indicating that the vast majority of our time is spent on scheduling and force evaluation, and not model regeneration. Comparing the timings for this example with the timings for a similar example reported by [Kaldor et al. 2008] (which

model	loops	segments	tolerance	contacts	updates	contact eval			overall (per 1/30s frame)		
						old	new	speedup	old	new	speedup
scarf	3,240	26,240	0.004	23,186	3.7%	254ms	**41.7ms**	**6.1x**	4m 10s	58.4s	4.3x
			0.04	23,475	0.46%		**30.9ms**	**8.2x**		50.2s	5.0x
			0.1	24,296	0.19%		**30.5ms**	**8.3x**		50.6s	4.9x
			0.3	29,037	0.04%		**32ms**	**7.9x**		50.8s	4.9x
sack	41,272	334,464	0.04	262,062	0.30%	3,063ms	**366ms**	**8.4x**	33m 55s	7m 27s	4.6x
afghan	54,340	438,485	0.04	365,305	0.47%	3,995ms	**600ms**	**6.7x**	44m 8s	10m 34s	4.2x
sweater	45,960	370,650	0.04	295,702	0.21%	3,665ms	**405ms**	**9.1x**	40m 16s	8m 6s	5.0x

Table 2: Model and scene statistics and timings. *All numbers and timings are an average number over 2s of simulation.*

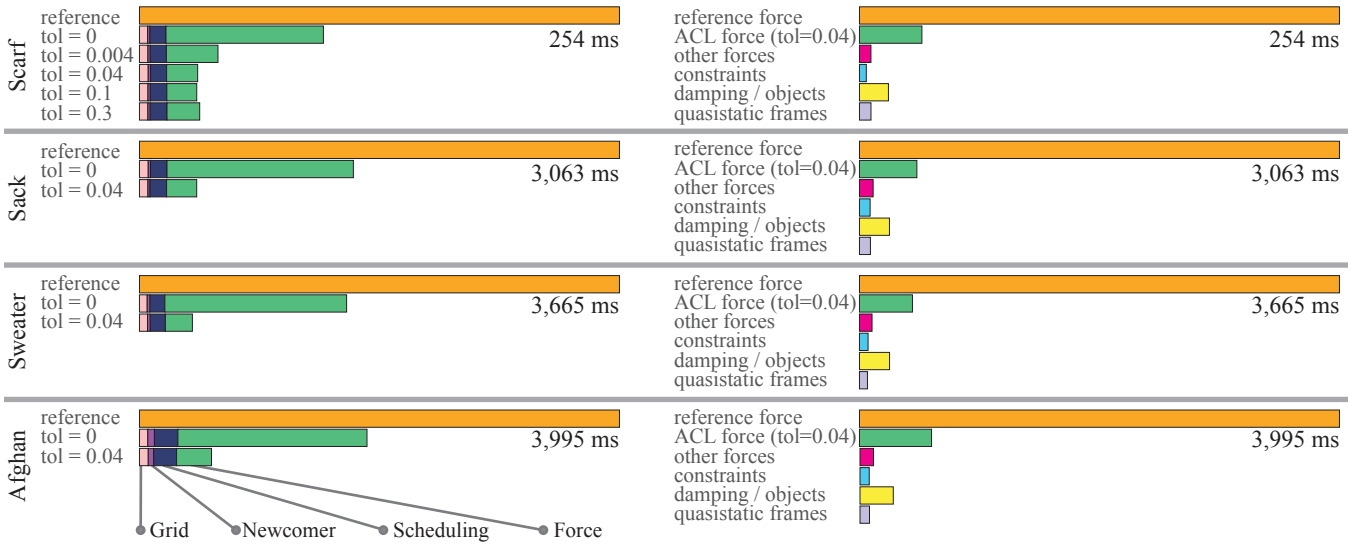


Figure 6: Performance comparison: *On the left is a performance breakdown of the various phases of our model, as compared to the time taken for the reference implementation. On the right is the breakdown of the costs of the phases of the simulator as a whole, again compared to the time taken for the force evaluation of the reference implementation, where ACL force is our adaptive contact linearization model.*

reported times of 10m 42s per frame) on similar hardware, our reference implementation takes 6m 52s per frame and our new model takes 1m 47s per frame.

However, the main benefit of our proposed ACL forces is the simulation of significantly larger yarn models, consisting of 41,000 to 54,000 knit loops (compare to the 3,240 loops in the scarf). As the sack (Figure 8) is filled with 90 rigid balls, our ACL forces localized model updates (exaggerated in red) primarily around where the spheres contact the cloth resulting in contact computation speedups from 7.5x to 9.4x (averaged over 5 frame intervals), and 4.3x to 4.8x overall. Our afghan (Figure 7) falls onto a sphere, causing high speed self-contacts across large portions of the cloth and large scale global deformations; this is the most challenging scenario for our model, resulting in speedups from 4.5x to 8.4x in contact computation and 3.3x to 4.7x overall. Finally, a wooden mannequin wearing a sweater walks forward (Figure 1), showing efficient simulation of character-sized garments in complex contact configurations; we achieve speedups from 7.5x to 10.5x in contact computation and 4.5x to 5.3x overall. The size of these models challenges the scalability of exploring empty space, but for instance on the sweater our scheduler limits the number of overall schedule entries being tracked to an average of 14 million, with on average only 1.2 million examined and 77,000 processed per timestep.

We note that in all scenes the percentage of model updates per timestep is extremely low, on the order of 0.5% per timestep, corresponding to more than 200 timesteps between invalidation for the mean contact set, or around three times per 1/30s frame. Thus, even though our timesteps are small, the temporal coherence is such that contact linearization and invalidation would still be effective even with a timestep 10x larger. In addition, even with relatively conservative tolerances the cost of model updates in an amortized sense was negligible, as seen in the cost for the various scarves; if larger timesteps were used and model updates became a performance bottleneck, the tolerance could be increased while still producing reasonable results.

7 Conclusion

We have demonstrated a method for speeding up contact force evaluations for yarn-based cloth models by breaking up the contact problem into a set of disjoint regions and adaptively constructing local models for each region to approximate the true force response. This results in typical speedups of 7x-9x over naïve force evaluation, which brings the cost of force evaluation in line with other phases of the simulation, while still maintaining similar (and in many cases, visually identical) motion.

We believe that this contact-level approach leads to many interesting future possibilities for both quality and performance improvements that are difficult to solve using naïve contact evaluation. In particular, we think that further additions may help address two of the most pressing problems in yarn-based simulation: modeling hysteresis and taking larger timesteps. Hysteresis and damping is a critical component to get right in order to achieve accurate cloth simulation, but it is currently addressed via damping of non-rigid motion. Adding plasticity in at the contact level is more physically plausible (due to fiber entanglement), and our contact set formula-

tion provides a primitive on which to model these effects, as in our simple approach of adding stiction springs.

Current timestep restrictions are due to two factors: the stiffness in the collision response, and, more crucially, the inability to detect and respond when the yarns slip through each other. This is a catastrophic simulator failure, since it can lead to the entire structure unravelling, and is currently addressed through small timestepping to ensure sufficient time for contact force response. Moving contact evaluation to a higher level (contact sets instead of collision points) should allow us to detect when this occurs. Ultimately, we hope that this will lead to the ability to adaptively timestep yarn-based models and automatically step down when the timestep becomes too aggressive and results in slip-through. Semi-implicit integration of contact groups might also address penalty-based stiffness to enable large timesteps [Baraff and Witkin 1998; Mirtich 2000].

Acknowledgments: This research was supported in part by the National Science Foundation (CCF-0702490), the NSF CAREER program (CCF-0347303, CCF-0652597), two Alfred P. Sloan Research Fellowships, and additional support from Intel, The Boeing Company, Pixar, Autodesk, NVIDIA, and Unilever. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

AN, S. S., KIM, T., AND JAMES, D. L. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM SIGGRAPH Asia 27*, 5, 164:1–11.

BARAFF, D., AND WITKIN, A. 1998. Large steps in cloth simulation. *ACM SIGGRAPH*, 43–54.

BARAFF, D., WITKIN, A., AND KASS, M. 2003. Untangling cloth. *ACM Trans. Graph. 22*, 3, 862–870.

BARBIČ, J., AND JAMES, D. 2005. Real-Time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Transactions on Graphics 24*, 3 (Aug.), 982–990.

BERGOU, M., WARDETZKY, M., HARMON, D., ZORIN, D., AND GRINSPUN, E. 2006. A quadratic bending model for inextensible surfaces. In *Fourth Eurographics Symposium on Geometry Processing*, 227–230.

BERGOU, M., WARDETZKY, M., ROBINSON, S., AUDOLY, B., AND GRINSPUN, E. 2008. Discrete elastic rods. *ACM SIGGRAPH*.

BERGOU, M., AUDOLY, B., VOUGA, E., WARDETZKY, M., AND GRINSPUN, E. 2010. Discrete viscous threads. *ACM SIGGRAPH*.

BERTAILS, F., AUDOLY, B., CANI, M.-P., QUERLEUX, B., LEROY, F., AND LÉVÊQUE, J.-L. 2006. Super-helices for predicting the dynamics of natural hair. *ACM Trans. Graph. 25*, 3 (August), 1180–1187.

BRIDSON, R., FEDKIW, R., AND JOHN ANDERSON. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM SIGGRAPH*, 594–603.

BRIDSON, R., MARINO, S., AND FEDKIW, R. 2003. Simulation of clothing with folds and wrinkles. *Symposium on Computer Animation 32*, 28–36.

CHADWICK, J. N., AN, S. S., AND JAMES, D. L. 2009. Harmonic shells: A practical nonlinear sound model for near-rigid thin shells. *ACM Transactions on Graphics 28*, 5 (Dec.), 119:1–119:10.

CHOI, K., AND KO, H. 2002. Stable but responsive cloth. *ACM SIGGRAPH*, 604–611.

CHU, L. 2005. *A Framework for Extracting Cloth Descriptors from the Underlying yarn Structure*. PhD thesis, University of California, Berkeley.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic real-time deformations using space and time adaptive sampling. *ACM SIGGRAPH*.

ENGLISH, E., AND BRIDSON, R. 2008. Animating developable surfaces using nonconforming elements. *ACM SIGGRAPH*.

ETZMUSS, O., KECKEISEN, M., AND STRASSER, W. 2003. A fast finite element solution for cloth modelling. In *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on*, 244–251.

FELIPPA, C. 2000. A systematic approach to the element-independent corotational dynamics of finite elements. *Center for Aerospace Structures Document Number CU-CAS-00-03, College of Engineering, University of Colorado*.

GAO, J., GUIBAS, L. J., AND NGUYEN, A. 2004. Deformable spanners and applications. *Proc. 20th ACM Symp. on Comp. Geom.*, 179–199.

GARG, A., GRINSPUN, E., WARDETZKY, M., AND ZORIN, D. 2007. Cubic Shells. In *2007 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 91–98.

GOLDENTHAL, R., HARMON, D., FATTAL, R., BERCOVIER, M., AND GRINSPUN, E. 2007. Efficient simulation of inextensible cloth. *ACM SIGGRAPH 26*, 3.

GOVINDARAJU, N. K., KNOTT, D., JAIN, N., KABUL, I., TAMSTORF, R., GAYLE, R., LIN, M. C., AND MANOCHA, D. 2005. Interactive collision detection between deformable models using chromatic decomposition. *ACM Trans. Graph. 24*, 3, 991–999.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. CHARMS: A simple framework for adaptive simulation. *ACM Transactions on Graphics 21*, 281–290.

GRINSPUN, E., HIRANI, A., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete shells. *Symposium on Computer Animation*, 62–67.

GUIBAS, L. 2004. Kinetic Data Structures. In *Handbook of Data Structures and Applications*, D. Mehta and S. Sahni, Eds. Chapman and Hall/CRC.

HARMON, D., VOUGA, E., SMITH, B., TAMSTORF, R., AND GRINSPUN, E. 2009. Asynchronous contact mechanics. *ACM SIGGRAPH*.

HUBBARD, P. 1995. Collision detection for interactive graphics applications. *IEEE Trans. Visualization and Computer Graphics 1*, 3, 218–230.

HUTCHINSON, D., PRESTON, M., AND HEWITT, T. 1996. Adaptive refinement for mass/spring simulations. In *Proceedings of the Eurographics workshop on Computer animation and simulation'96*, Springer-Verlag New York, Inc., 45.

KALDOR, J., JAMES, D. L., AND MARSCHNER, S. 2008. Simulating knitted cloth at the yarn level. *ACM SIGGRAPH*.

KAWABATA, S., NIWA, M., AND KAWAI, H. 1973. The finite deformation theory of plain-weave fabrics part I: The biaxial-deformation theory. *Journal of the Textile Institute 64*, 21–46.

KHAREVYCH, L., MULLEN, P., OWHADI, H., AND DESBRUN, M. 2009. Numerical coarsening of inhomogeneous elastic materials. *ACM Trans. Graph. 28*, 3, 1–8.

KIM, T., AND JAMES, D. L. 2009. Skipping steps in deformable simulation with online model reduction. *ACM Trans. Graph. 28*, 5, 1–9.

KING, M., JEARANAISILAWONG, P., AND SCORATE, S. 2005. A continuum constitutive model for the mechanical behavior of woven fabrics. *International Journal of Solids and Structures 42*, 3867–3896.

MIRTICH, B. 2000. Timewarp rigid body simulation. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, 193–200.

MÜLLER, M., AND GROSS, M. 2004. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, 239–246.

MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. 2002. Stable real-time deformations. In *ACM SIGGRAPH Symposium on Computer Animation*, 49–54.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. *ACM SIGGRAPH 24*, 3, 471–478.

NESME, M., KRY, P. G., JEŘÁBKOVÁ, L., AND FAURE, F. 2009. Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph. 28*, 3, 1–9.

PAI, D. 2002. STRANDS: Interactive simulation of thin solids using Cosserat models. *Eurographics 21*, 347–352.

PROVOT, X. 1995. Deformation constraints in a mass-spring model to describe rigid cloth behavior. *Proc. Graphics Interface '95*, 147–154.

PROVOT, X. 1997. Collision and self-collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation '97: Proceedings of the Eurographics Workshop in Budapest, Hungary, September 2-3, 1997*, Springer, 177.

RIVERS, A. R., AND JAMES, D. L. 2007. FastLSM: Fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph. 26*, 3, 82.

SPILLMANN, J., AND TESCHNER, M. 2008. An adaptive contact model for the robust simulation of knots. *Eurographics 27*, 497–506.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. *Computer Graphics 21*, 205–214.

TESCHNER, M., KIMMERLE, S., HEIDELBERGER, B., ZACHMANN, G., RAGHUPATHI, L., FUHRMANN, A., CANI, M., FAURE, F., MAGNENAT-THALMANN, N., STRASSER, W., AND VOLINO, P. 2005. Collision detection for deformable objects. In *Computer Graphics Forum*, vol. 24, Blackwell Publishing, 61–81.

THEETTEN, A., GRISONI, L., DURIEZ, C., AND MERLHIOT, X. 2007. Quasi-dynamic splines. In *Proc. ACM Symposium on Solid and Physical Modeling '07*.

VILLARD, J., AND BOROUCHAKI, H. 2005. Adaptive meshing for cloth animation. *Engineering with Computers 20*, 4, 333–341.

VOLINO, P., AND THALMANN, N. M. 2000. Implementing fast cloth simulation with collision response. In *Proc. Computer Graphics International*, 257–266.

VOLINO, P., MARTIN COURCHESNE, AND MAGNENAT-THALMANN, N. 1995. Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proc. SIGGRAPH '95*.

ZENG, X., TAN, V. B. C., AND SHIN, V. P. W. 2006. Modelling inter-yarn friction in woven fabric armor. *International Journal for Numerical Methods in Engineering 66*, 1309–1330.

A Non-Zero-Twist Reference Frames

We follow the notation of Bergou et al. [2008]. We eliminate the requirement that the reference frame $(\mathbf{u}^i, \mathbf{v}^i)$ have zero twist, and instead update each $\mathbf{u}^i$ at time $t+h$ by parallel transporting the previous $\mathbf{u}^i(t)$ through time (i.e. by parallel transporting $\mathbf{u}^i$ from the vector $\mathbf{e}^i(t)$ to $\mathbf{e}^i(t+h)$). We denote the twist from $(\mathbf{u}^i, \mathbf{v}^i)$ to $(\mathbf{u}^{i+1}, \mathbf{v}^{i+1})$ as $\hat{\theta}^{i+1}$. This twist can be computed on each timestep by computing the angle between $P(\mathbf{u}^i)$ and $\mathbf{u}^{i+1}$, i.e., the twist between the parallel transported $\mathbf{u}^i$ (which will have zero twist relative to $\mathbf{u}^i$) and $\mathbf{u}^{i+1}$, taking care to handle relative twists greater than 2π properly.

Next, we must redefine the twisting and bending energy (and respective derivatives) to include these changes. The modified twisting energy is

$$E_{\text{twist}} = \sum_{i=1}^{n} k_{\text{twist}} \frac{(\theta^i - \theta^{i-1} - \hat{\theta}^i)^2}{\bar{\ell}_i}$$

It is straightforward to compute $\frac{\partial E_{\text{twist}}}{\partial \theta^i}$ because $\frac{\partial \hat{\theta}^j}{\partial \theta^i} = 0$ for all i, j. For $\frac{\partial E_{\text{twist}}}{\partial \boldsymbol{x}_i}$, we obtain

$$\frac{\partial E_{\text{twist}}}{\partial \boldsymbol{x}_i} = \sum_{j=0}^{n} \frac{\partial E_{\text{twist}}}{\partial \theta^j} \frac{\partial \theta^j}{\partial \boldsymbol{x}_i} + \frac{\partial E_{\text{twist}}}{\partial \hat{\theta}^j} \frac{\partial \hat{\theta}^j}{\partial \boldsymbol{x}_i}$$

where $\frac{\partial \theta^j}{\partial \boldsymbol{x}_i} = 0$ for all i, j because each frame is independently parallel transported through time; $\frac{\partial \hat{\theta}^j}{\partial \boldsymbol{x}_i}$ is not necessarily zero since it depends on $P(\mathbf{u}^i)$ and $\mathbf{u}^{i+1}$. However, because $\mathbf{u}^i$ and $\mathbf{u}^{i+1}$ are updated via parallel transport through time (i.e. with zero twist), this corresponds precisely to the gradient of holonomy of $P(\mathbf{u}^i)$ relative to $\mathbf{u}^i$ (§6 of [Bergou et al. 2008]):

$$\frac{\partial \hat{\theta}^i}{\partial \boldsymbol{x}_{i-1}} = \frac{(\kappa \mathbf{b})_i}{2|\bar{e}^{i-1}|}$$

$$\frac{\partial \hat{\theta}^i}{\partial \boldsymbol{x}_{i+1}} = -\frac{(\kappa \mathbf{b})_i}{2|\bar{e}^{i}|}$$

$$\frac{\partial \hat{\theta}^j}{\partial \boldsymbol{x}_i} = \begin{cases} -\left(\frac{\partial \hat{\theta}^i}{\partial \boldsymbol{x}_{i-1}} + \frac{\partial \hat{\theta}^i}{\partial \boldsymbol{x}_{i+1}}\right), & i = j \\ 0, & \text{otherwise.} \end{cases}$$

As for the bending energy, the definition itself does not change, but its derivatives do. Because each $\mathbf{u}^i$ is parallel transported through time, there is no need to account for the variation in the Bishop frame when computing the derivative. As a result, the gradient of the material-frame curvature (eq 11 in [Bergou et al. 2008]) is now

$$\nabla_i \omega_k^j = \begin{pmatrix} (\mathbf{m}_2^j)^T \\ -(\mathbf{m}_1^j)^T \end{pmatrix} \nabla_i (\kappa \mathbf{b})_k.$$

Because $\nabla_i (\kappa \mathbf{b})_k$ is nonzero only for $k-1 \le i \le k+1$, the gradient of the material-frame curvature is nonzero only for $k-1 \le i \le k+1$, and so the summation over all points in the bending energy is now unnecessary—summation merely needs to occur over the three-vertex stencil for each bending element.

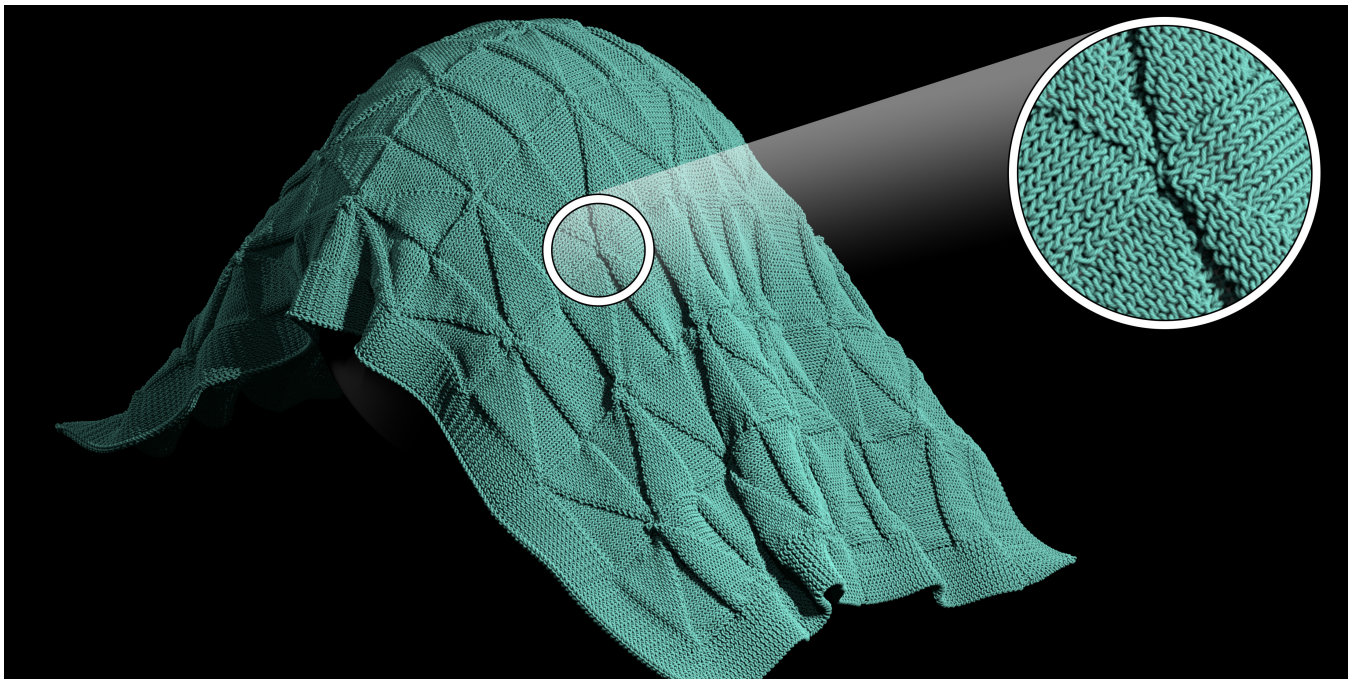


Figure 7: Afghan: *Our adaptive linearized contact model performs well even for cloth undergoing large global and local deformations.*

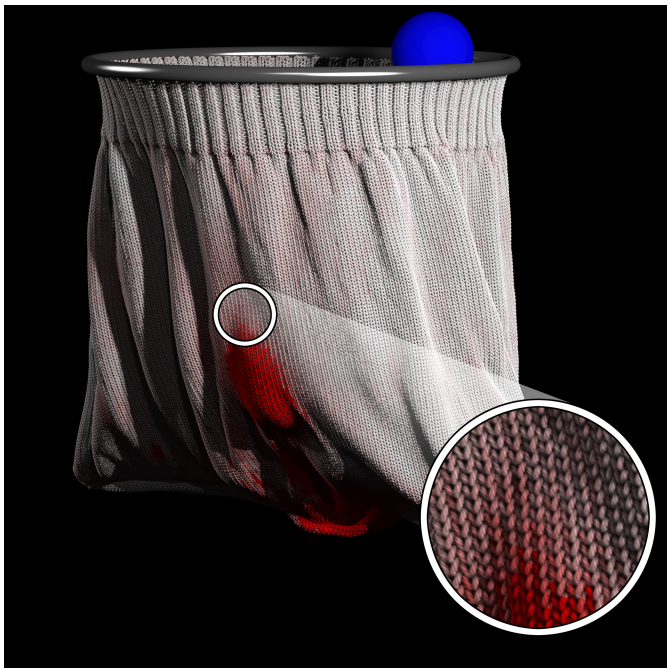
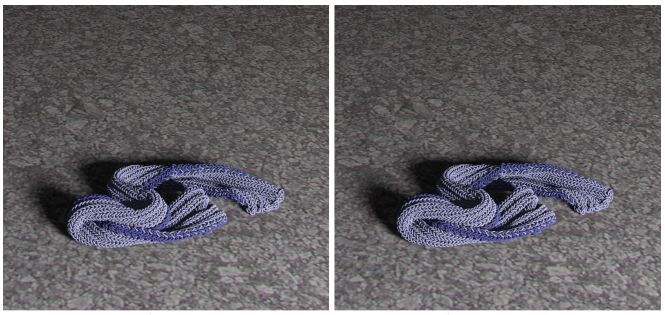


Figure 8: Sack: *Model updates are focused on the regions undergoing deformation, as seen here when struck by falling spheres. Pure red corresponds to* ≥ 13 *updates over the 1/30s frame (540 timesteps)*



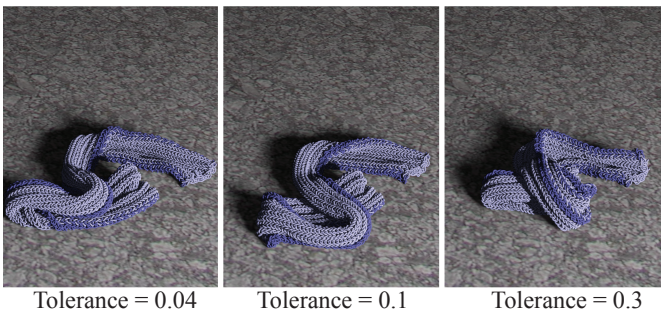


Figure 9: Scarf Comparisons: *A scarf falling on a flat plane for a variety of tolerances. Our model faithfully captures the same qualitative movement even for aggressive tolerances.*