Animating Developable Surfaces using Nonconforming Elements

Elliot English
eenglish@cs.ubc.ca
University of British Columbia

Advisor: Robert Bridson

May 1, 2008

Figure 1: *Frames from an animation of exactly developable cloth, with complex collisions. The underlying simulation mesh only bends; it has zero stretch or compression in every direction.*

1 Introduction

Many deformable surfaces, ranging from most types of cloth to paper [15, 4] and stiffer materials, are well approximated as *developable*: they bend out-of-plane but do not visibly stretch or compress in-plane. Even for materials which do allow some in-plane deformation, e.g. small amounts of shearing in a fabric relative to the warp and weft directions, if the simulator cannot handle the developable limit there are bound to be numerical problems as users attempt to approach it. Thus in this thesis I focus attention on the fully developable case, imposing zero in-plane deformation as a hard constraint, though of course the technique present here is easily generalized to stretchy or shearable materials (though, note that if stretchy enough that existing simulators can accurately handle them, the constant factor overheads of the presented method will make it uncompetitive).

Unfortunately, standard graphics simulators break down precisely at this limit. For example, for a triangle mesh with the usual piecewise linear elements, developability implies that each triangle remain rigid. For any nontrivial bending this constraint must be violated: the mesh can essentially only crease along straight lines already present in the mesh as edges: it *locks*.

The general phenomena of locking, i.e. the inability of a given finite element space to approximate solutions [5, 12], was recently brought to light in graphics for volume-conserving volumetric simulations by Irving et al.[13]. In the developable surface case, Liu et al.'s rigorous analysis shows that a general n-triangle conforming mesh only has $O(\sqrt{n})$ degrees of freedom[16], with significant mesh-dependent artifacts. Similar arguments show that quad meshes with bilinear elements suffer from the same locking problem, as do many higher order polynomial elements. For stiff but not fully constrained cloth models, e.g. where edges may change their length slightly, locking manifests as a spurious increased resistance to bending, proportional to the in-plane stiffness rather than the true bending stiffness.

The classic solution to the locking problem for volume-conserving deformations is to use *nonconforming* elements [5]. Rather than reduce the number of constraints by averaging over larger regions as Irving et al. propose, finite element practitioners traditionally increase the number of variables, by putting

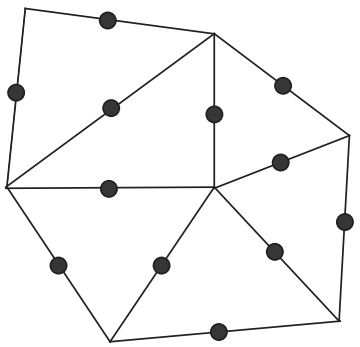

Figure 2: *Schematic of nonconforming variables, located at midpoints of edges between triangles. While continuous at these points, the surface may be discontinuous along the rest of each edge.*

the variables at the midpoints of edges (in 2D) or faces (in 3D).[1] I adopt this approach for the deformable surface case by putting the position variables at the midpoints of the edges rather than at the vertices of the triangles in the mesh (see figure 2). This now gives $3e \approx 9v$ variables, and to make each linear triangle rigid only implies $3t \approx 6v$ constraints, leaving approximately $3v$ true degrees of freedom for bending—allowing the method to accurately approximate developable surfaces. The first part of this thesis gives the details on this approach: how to enforce developability, special treatment of boundary elements, and a crude model for bending forces.

However, the discrete surfaces that now arise are nonconforming, i.e. no longer necessarily continuous: adjacent triangles only have to meet at the midpoint of the common edge, not necessarily at vertices. This clearly poses a problem for robust collision processing and rendering. I therefore propose the coupling of a "ghost" conforming mesh (with the usual vertex variables) to the simulation, used solely for collisions and rendering.

Finally, since the method imposes developability as a hard constraint, turning to time integration of constrained mechanics. The usual schemes in graphics unfortunately suffer from strong numerical damping with nonlinear constraints, as energy is erroneously transferred to constrained modes and projected out. I propose a new second order accurate multistep method, based on BDF2 and simple position-based constraint projection. This both reduces numerical damping and speeds projection (since the surface remains closer to the constraint manifold), without need for stabilization or velocity projection.

[1] This may be identified as the lowest order Crouzeix-Raviart element.

2 Previous Work

Cloth simulation has a long history within computer graphics; works highlighted here are just a sampling of the relevant papers from the standpoint of the developable limit.

Provot[18] worked with mass-spring models, introducing a loose constraint on edges to not deform by more than 10% with a simple Gauss-Seidel iteration. Bridson et al.[7] demonstrated improved buckling behavior if edges were (loosely) constrained to not compress at all, just stretch. The critical aspect of this approach, though not identified at the time, was that triangles were left with some freedom to deform, fortuitously avoiding the locking problem. Of course, this can't realistically handle the many materials which more severely limit strain, and the Gauss-Seidel constraint iteration tends to induce mesh-dependent artifacts. This paper also makes use of the robust collision processing algorithm (for conforming triangle meshes) developed by Bridson et al.[6] from earlier work by Provot [19].

Baraff and Witkin[2] instead proposed a semi-implicit Backwards Euler integrator to avoid the stability time step restriction plaguing explicit time integration of stiff models. Later authors argued that the strong numerical damping present in Backwards Euler was responsible for the fairly smooth appearance of the Baraff and Witkin results; I suspect a large share of the problem was locking, as very stiff in-plane forces brought the material model close to developable, causing spurious numerical resistance to bending.

Choi and Ko[8] introduced the second order accurate BDF2 method to cloth simulation, which features much reduced damping yet still has stiff decay [1], of crucial importance for dealing with stiff systems.[2] They also fortuitously avoided locking with their implicit model of the buckling instability, allowing edges in their model to compress easily while still offering stiff resistance to stretch. (This can also be viewed as using biphasic springs, with lower resistance to compression than stretch.) This is an attractive solution for much of the large-scale motion of the cloth, but the buckling model causes small-scale details to remain implicit—i.e. not visible in the simulation mesh. Attempts at procedurally adding in the missing detail have met with mixed success (e.g. [22, 14, 21]).

More recently Goldenthal et al.[10] demonstrated an effective approach to constraining a quad-dominant cloth mesh to zero deformation along the warp and weft directions. They avoid locking by requiring most of the mesh to use quad elements, and by not constraining shearing: this leaves enough degrees of freedom to accurately and beautifully capture many fabrics of interest, but it cannot be extended to the developable no-shear limit or to triangle meshes. The new constrained mechanics time integration scheme presented hre is a multistep extension of Goldenthal et al.'s fast projection method.

Bergou et al.[3] introduced the nonconforming elements used here in the

[2]By contrast, implicit symplectic integrators, such as certain Newmark schemes, cannot possess stiff decay and thus exhibit objectionable temporal aliasing of high frequency modes into low frequency modes when using large time steps.

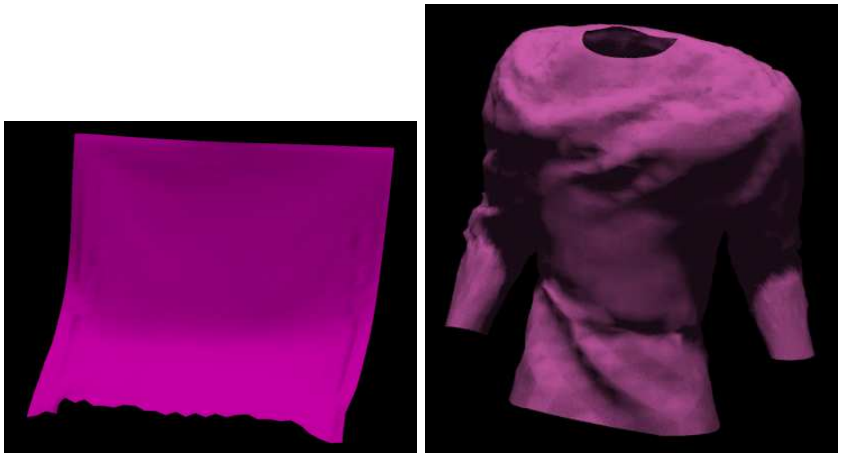

Figure 3: *On the left is a square of developable surface pinned at two corners: we enforce zero in-plane deformation to a relative error tolerance of* 10^{-4}*, giving virtually no sag along the top edge. Extra constraints ensure realistic behavior at the boundaries. To the right is a irregularly meshed nonplanar shirt worn by a moving character.*

context of deriving a compact stencil for bending forces on conforming meshes; this work generalizes the elements to use them for in-plane dynamics to solve locking.

Liu et al.[16] imposed developability as a constraint on conforming triangle meshes, proving that n triangles give you $O(\sqrt{n})$ degrees of freedom under isotropic conditions. In a regularly spaced plane, these degrees of freedom manifest themselves as the bending angles along parallel edges in both horizontal and vertical directions as well as the one diagonal direction. The simulation algorithm is based upon the integration of these degrees of freedom, explicitly defining them by iteratively walking through the mesh and assigning degrees of freedom to under constrained features as they are visited. The crucial difference between this model and previous methods discussed is that the model by definition is developable for any values of these degrees of freedom. Locking is inherent to this model, independently of the mesh resolution, with artifacts remaining even in the limit. Additionally, this geometric model makes handling collisions extremely difficult while maintaining the developable constraint.

3 Simulation Model

3.1 A Nonconforming Element Discretization

Beginning with a regular triangle mesh in parameter or "object" space, with the midpoint of each edge i at parameter space position p_i as in figure 2. Each edge variable also has a world space position x_i, a velocity v_i, etc. Within a

triangle with edges i, j, and k, the variables are extended with linear interpolation/extrapolation: e.g. from geometric similarity the world space position of the vertex located at the corner opposite edge i is $x_j + x_k - x_i$. This can also be phraseed in terms of piecewise linear basis functions $\{\phi_i\}$, where $\phi_i(p)$ is 1 all along edge i and zero at the midpoints of all other edges: $x(p) = \sum_i x_i \phi_i(p)$.

The mass m_i associated with edge i is simply a third of the mass of the sum of the masses of the incident triangles; these can be assembled into a diagonal mass matrix M, with each mass repeated three times. Newton's law is then $DA^2/dt^2 = M^{-1}F$, where F is a vector of the net forces on each edge.

For a regular elastic material, one could use the usual Galerkin finite element discretization (see Brenner and Scott[5] for example), integrating gradients of the nonconforming basis functions over each triangle as appropriate to get a stiffness matrix, but—crucially—avoiding integrating over the jump discontinuities on the edges between triangles.

However in the developable limit, an equivalent but simpler formulation is possible. The deformation gradient in each triangle is the gradient of world space position w.r.t. parameter values. For linear elements, this is constant in each triangle; to avoid in-plane deformation, this gradient matrix must be orthogonal, i.e. the world space pose of each triangle must be a rigid transformation of the parameter space pose. A triangle is rigid if and only if the distance between any two edge midpoints remains constant, giving three constraints per triangle of the form

$$c_{ij}(x) = \|x_i - x_j\|^2 - d_{ij}^2 = 0 \tag{1}$$

where d_{ij} is the parameter space distance between edge midpoints i and j. These constraints are then assembled into one column-vector-valued function $C(x)$.

This is now a discrete constrained mechanics problem, with Lagrange multiplier constraint forces of the form

$$F_c = \left(\frac{\partial C}{\partial x}\right)^T \lambda = J^T \lambda \tag{2}$$

where $J = \partial C/\partial x$ is the Jacobian of the constraint function and λ is a vector containing one Lagrange multiplier per constraint. In section 5.1 I will discuss methods for integrating this motion.

While this model is robust when the motion of the triangles along the edges of the mesh is prescribed, for more typical free boundary situations artifacts do arise: the per-triangle rigidity constraints aren't quite enough here. For example, a corner triangle with two boundary edges—and thus only one edge shared with another triangles—is free to rotate arbitrarily, independent of the orientations of nearby triangles. Even if a triangle only has one boundary edge, and is connected to the rest of the mesh at two edge midpoints, it is also free to spin around one axis independently of the orientations of nearby triangles. In fact, even when restricted to planar motion the model fails: the only planar developable motions are globally rigid, yet this model allows additional deformations.[3] In the planar elasticity context, this is a known instability due to free

[3]This may be familiar from condensed matter physics under the topic of Kagome Lattices.

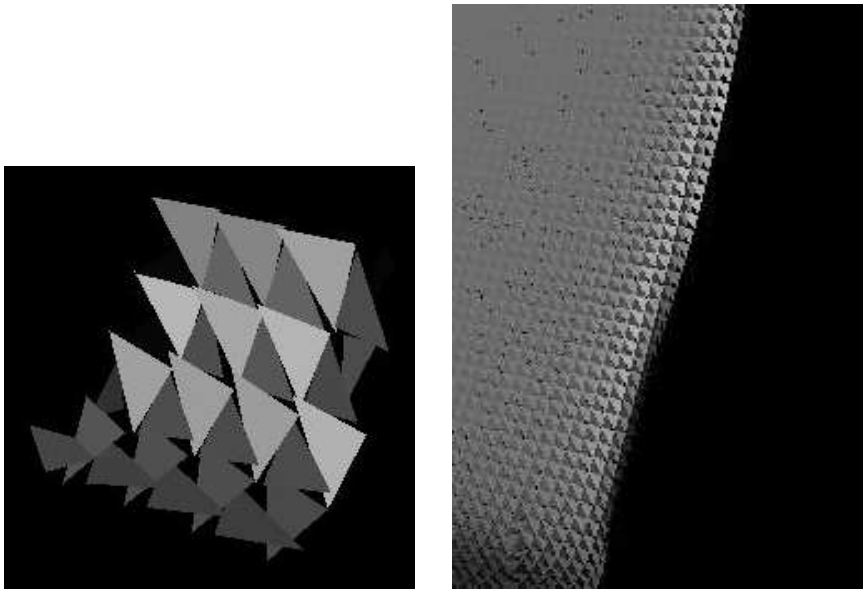

Figure 4: *Without additional boundary constraints underconstrained elements have spurious degrees of freedom.*

traction boundary conditions (as opposed to displacement boundary conditions, i.e. prescribed positions): see e.g. Falk[9].

Thus further boundary constraints are needed: I require that for each boundary vertex the corresponding vertices on the incident nonconforming elements conform. As a result these elements are also C^0 along their shared edges, have their orientations properly coupled, and no longer have spurious rotational freedom: the outer "ribbon" of triangles, i.e. all those that touch the boundary, are forced to be conforming while the interior of the mesh is still free to be nonconforming. Figure 3, showing a rectangle pinned at two corners, illustrates the robustness of this approach. This also resolves the planar deformation problem, since the only solution for the conforming outer ribbon is globally rigid motion, which induces a rigid position boundary condition on the interior nonconforming elements which in turn has been proven to only have the expected rigid motion as a solution [5].

3.2 Bending Forces

While discretization of bending forces is not the focus of this paper, obviously conforming mesh methods based on vertex unknowns can't be directly applied. Traditional finite element approaches for plate or shell problems (or the biharmonic model problem) include a different nonconforming scheme, the Morley element: a piecewise quadratic with nodal variables at triangle vertices and normal derivatives variables at edge midpoints. However, the piecewise lin-

ear element presented above can't directly be used in the traditional Galerkin framework for bending, since the associated bilinear form uses second derivatives which, for linear elements, are exactly zero.

This is the case for conforming linear elements too, of course, and just as graphics researchers have done in that case, slightly more exotic mixed finite element methods can be used. Wardetzky et al.[23] recently published an alternative derivation based on discrete geometric principles, albeit restricted to the conforming mesh case: while they use the same linear edge-based elements, they only use the conforming subspace, mapping edge unknowns back to vertex variables. Like Wardetzky et al. and earlier work by Bergou et al.[3] the highly nonlinear bending energy is reduced to a simple quadratic form by exploiting developability—though whereas for earlier papers this was a rough approximation, the nonconforming surfaces here will be exactly developable and thus no approximation is necessary. That is to write the bending potential energy as:

$$E_b = \frac{k_b}{2} \iint \|\nabla^2 x\|^2 dp \tag{3}$$

where the integral is taken in parameter space, k_b is the bending stiffness parameter, and $\nabla^2 x$ is the Laplacian of world space positions with respect to parameter space values. Introduce a secondary variable u as a stand-in for the Laplacian of x:

$$E_b = \frac{k_b}{2} \iint \|u\|^2 dp, \qquad u = \nabla^2 x \tag{4}$$

Now discretize both x and u with the linear nonconforming elements given above, represented with nodal basis functions $\{\phi_i\}$:

$$x(p) = \sum_i x_i \phi_i(p), \qquad u(p) = \sum_i u_i \phi_i(p) \tag{5}$$

Substituting these in to the mixed form of the bending energy, equation 4, and integrating the $u = \nabla^2 x$ equation with ϕ_j with the usual integration by parts and neglecting boundary terms for now, gives after rearranging:

$$\begin{gathered} E_b = \frac{k_b}{2} \sum_i \sum_j \left(\iint \phi_i \phi_j dp \right) u_i \cdot u_j \\ \sum_i \left(\iint \phi_i \phi_j dp \right) u_i = -\sum_i \left(\iint \nabla \phi_i \cdot \nabla \phi_j dp \right) x_i \end{gathered} \tag{6}$$

These integrals are evaluated per triangle, again avoiding the discontinuous jumps along edges. Let K be the stiffness matrix for the Laplacian, $K_{ij} = \iint \nabla \phi_i \cdot \nabla \phi_j dp$, and W be the mass matrix, $W_{ij} = \iint \phi_i \phi_j dp$, which can be verified to be diagonal. Arrving at:

$$E_b = \tfrac{1}{2} k_b u^T W u, \qquad Wu = -Kx \tag{7}$$

Eliminating u gives the final discrete bending energy:

$$E_b = \tfrac{1}{2} k_b x^T K^T W^{-1} K x \tag{8}$$

Elastic bending forces on the edges are simply the gradient of the bending potential energy:

$$F_b = Bx \tag{9}$$

where $B = k_b K^T W^{-1} K$. Note that B is necessarily symmetric positive semi-definite, with a null-space that includes all unbent (linear) configurations of the mesh, as expected. Implicit time integration using B is straightforward.

The issue of bending boundary conditions is avoided by simply zeroing out rows of K corresponding to boundary variables before forming B, which in practice seems to give plausible behavior.

4 Collisions and Rendering

4.1 Ghost Mesh Coupling

The nonconforming surface may have jump discontinuities along edges, apart from at the midpoints. Rendering this surface directly shows undesirable cracks between triangles. Similarly if the cloth collision algorithm were run on it potentially fatal scenarios could arise as the exposed edges of the nonconforming elements would allow the elements to tangle with or pass through one another.

One solution to these problems is to maintain a ghost conforming mesh (with vertex-based positions, see figure 5) of the same topology as the nonconforming mesh, initialized with the same geometry. Each time step begins by computing candidate new edge positions x_n and velocities v_n for the nonconforming mesh based on non-collision forces (gravity, internal constraints, bending, etc.). These are then transfered to candidate new vertex positions x_c and velocities v_c for the conforming mesh, by taking the average at each vertex of the nonconforming values extrapolated from all incident triangles. This is then represented with an averaging matrix A:

$$x_c = Ax_n \tag{10}$$

These give candidate conforming mesh trajectories, from the positions at the end of the last time step x_c^{old} (which are guaranteed to be non-interpenetrating) to the new conforming positions x_c, with new velocities $v_c = Av_n$. This mesh is then fed into a standard cloth collision code [6] to solve for the final conforming positions x_c' and velocities v_c', which should be non-interpenetrating. This new intersection-free conforming mesh is saved for later rendering and collision processing in the following step.

Once operations on the conforming mesh are complete the nonconforming mesh is updated to its final positions x_n', coupling the effect of collisions back into the main simulation. Taking a Lagrange multiplier form for the nonconforming correction:

$$x_n' = x_n + A^T \lambda \tag{11}$$

where λ is chosen so that averaging the final nonconforming positions x_n' back to the conforming mesh vertices returns the final conforming positions:

$$Ax_n' = x_c' \tag{12}$$

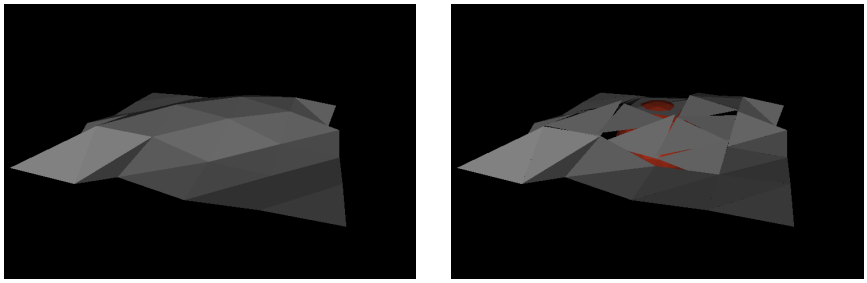

Figure 5: *The ghost mesh closely tracks the nonconforming mesh while maintaining a penetration free state.*

This gives a simple symmetric positive definite linear system to solve, which simplifies to:

$$AA^T\lambda = x_c' - x_c \tag{13}$$

Note that the coefficient matrix AA^T is constant, thus Cholesky factorization can be run once and then very efficiently solve this at every subsequent time step. Also note that if no collisions occur, the nonconforming positions are not modified at all, as one would expect. Finally the process concludes by updating the nonconforming velocities in a consistent manner:

$$v_n' = v_n + \frac{x_n' - x_n}{\Delta t} \tag{14}$$

also noting here that in the absence of collisions the velocities are not modified at all.

In summary, after advancing the nonconforming mesh, I update the conforming mesh by averaging vertices, run collision handling, and then use Lagrange multipliers to update the nonconforming mesh. Figures 1 and 9 show the robustness of this scheme, guaranteeing an intersection-free conforming mesh that closely tracks the simulation even in complex self-folding scenarios.

While this method has produced reasonable results, there are two chief issues I would like to resolve in future work. The first is that the simple averaging to get the conforming mesh often doesn't provide as smooth a surface as might be desired, as might be expected since the sampling density of the conforming mesh is in fact lower, and that the conforming mesh is only approximately developable (though its motion does show the particular developable dynamics of the underlying nonconforming mesh). One possible solution is to use edge subdivision schemes (e.g. [17]) which may help here. The other issue is that collisions are handled after internal dynamics, introducing an $O(\Delta t)$ splitting error which can perturb developability. I have not noticed any obvious artifacts stemming from these issues, but in principle they could be a concern especially in severe collision scenarios with large time steps.

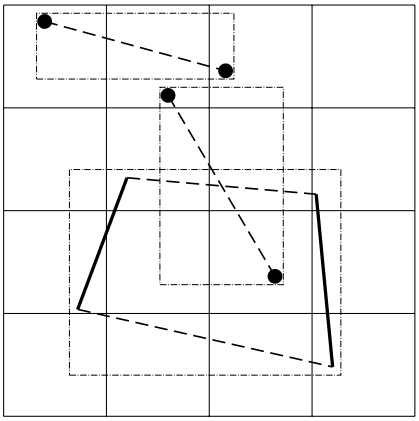

Figure 6: *A 2d dimensional hash grid acceleration structure showing two swept vertices and one swept edge. Note that the bounding boxes of the higher vertice and the edge are disjoint with respect to the grid cells the occupy, allowing for quick elimination.*

4.2 Collision Handling

Given an arbitrary triangular mesh two distinct types of collisions can occur, these being triangle-vertex and edge-edge collisions. The assumption is then made the motion of each vertex over the course of a time step is linear, allowing collisions to then be found by linearly sweeping each feature. In 4 dimensional this represents vertices as lines, edges as bilinear patches and triangles as prisms with bilinear patch sides. The problem is then reduced to finding the roots of the signed volume of the tetrahedron formed by the four vertices as a function of time. Additionally, at each candidate time that lies within the current time step, proceeding in order from the earliest time, the geometry is checked for a proximity of 0 and not merely coplanarity.

As per [6] the cloth collision handling algorithm procedes in three stages.

The first is a repulsion force stage intended to maintain a small separation distance between all of the cloth geometry at the current position (see figure 7). Given two features that have a proximity of less than the separation distance, the updated relative normal velocity becomes

$$v_n^1 = max(v_n^0 \cdot n, \frac{d_0 - d}{\Delta t T^0})n \tag{15}$$

where d_0 is the desired separation distance, d is the current separation distance, n is the normal direction point in the separating direction of the relative velocity and v_n^0 is the current relative normal velocity. T^0 is a constant specifying the time for the features to reach the separation distance. Additionally during this phase friction impulses are applied by The updated relative tangential velocity is defined as

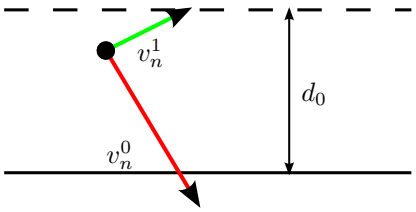

Figure 7

$$v_t^1 = max(0, 1 - \mu \frac{|v_n^1 - v_n^0|}{|v_0^t|}) v_t^0 \tag{16}$$

where μ is the coefficient of friction.

In the second stage swept collision testing to used to identify pairs that collide over the time interval between the current and previous steps. Upon detecting a collision, an impulse is applied to the geometry setting the relative normal velocity to zero, $v_n = 0$.

In the final stage, in order to ensure that no collisions are present once collision processing is finished, rigid impact zones are calculated. This proceeds by checking for collisions as in the second step, except that when a collision is detected the impact zones for each of the four vertices involved are merged and the new rigid motion is calculated and applied to each vertex in this new impact zone. The rigid motion is found by adding up the inertia matrices of each vertex and their momentums about the center of mass and then calculating the new rigid velocities for each vertex (see [6, 19] for details). Initially, each vertex is in their own impact zone, with the algorithm continuing until either there are no more collisions or all the vertices are in the same impact zone, guaranteeing no collisions in the final solution since the vertices no longer move relative to one another.

In order to accelerate feature proximity calculation and collision detection a spatial subdivision structure is used to quickly eliminate geometry that neither collides or is in close proximity. The major requirement of this structure is that it can be updated in constant time as the space occupide by the each swept volumes can change during collision processing. Tradional hierarchical structures such as oriented bounding box trees [11] require expensive recomputation of principle axes at every update if a tight bound is to be maintained. Instead, I implemented a three dimensional hash grid in to which geometry is scanned based up their bounding box (see figure 6). Additionally, when querying the grid for intersections with a bounding box, a procedure known as mail-boxing is used in order to prevent features from being redundently added to the candidate list. Simply by assigning a query id, and tracking the last query id by feature, each feature can then be added only once to the candidate list per query.

In the collision repulsion stage only the current positions are used when determining proximity so only the bounding boxes based upon current positions

are scanned into the acceleration grid at this time. In the collision impulse and rigid impact zones stages, first a list of all possible triangle-vertex and edge-edge collision candidates are assembled into a queue. This queue is then traversed and upon handling a collision all affected neighboring geometry is then rechecked for collisions and candidates added to the end of the queue. One final optimization is that by assuming most if not all collisions are handled by the collision impulses stage, the remaining candidate list is then passed to the rigid impact zone stage since all entries not on the list are known to not be intersecting given the current state.

5 Time Integration of Constraints

5.1 Reduced Numerical Dissipation Time Integration

Goldenthal et al.[10] introduced "fast projection", a particularly attractive first order accurate time integrator for constrained mechanics. This may be derived, heuristically at least, as the limit of Backwards Euler applied to a system where hard constraints $C(x) = 0$ are replaced with very stiff elastic forces $F_c(x) = -kJ^T(x)C(x)$, the gradient of the potential $\frac{1}{2}k\|C(x)\|^2$. Backwards Euler to get to time step $n+1$ is:

$$\begin{aligned} x^{(n+1)} &= x^{(n)} + \Delta t v^{(n+1)} \\ v^{(n+1)} &= v^{(n)} + \Delta t M^{-1}(F_a - kJ^T C) \end{aligned} \tag{17}$$

where F_a are the applied non-constraint forces (gravity and bending), and J and C are evaluated at $x^{(n+1)}$. Eliminating velocity:

$$x^{(n+1)} + k\Delta t^2 M^{-1} J^T C = x^{(n)} + \Delta t v^{(n)} + \Delta t^2 M^{-1} F_a \tag{18}$$

Taking the limit as $k \to \infty$ can be shown, at least for constraints close enough to linear in a local neighborhood, to give the fast projection algorithm:

$$\begin{aligned} x_0 &= x^{(n)} + \Delta t v^{(n)} + \Delta t^2 M^{-1} F_a \\ x^{(n+1)} &= \text{project}(x_0) \end{aligned} \tag{19}$$

Here x_0 is a predicted position, and the second step projects this onto the constraint manifold, so that $C(x^{(n+1)}) = 0$. Once the new $x^{(n+1)}$ is found, the new velocity can be determined from the Backwards Euler step $x^{(n+1)} = x^{(n)} + \Delta t v^{(n+1)}$.

While elegant and effective, this algorithm can suffer from numerical damping when the constraints are nonlinear. From a velocity perspective, even if the time n velocity is tangent to the constraint manifold at $x^{(n)}$, the tangent space changes at $x^{(n+1)}$ so the velocity components now normal to the constraint are projected out, for a net loss of kinetic energy.

I therefore propose a multistep version of this algorithm, based instead on the second order accurate multistep method BDF2, which similarly has stiff

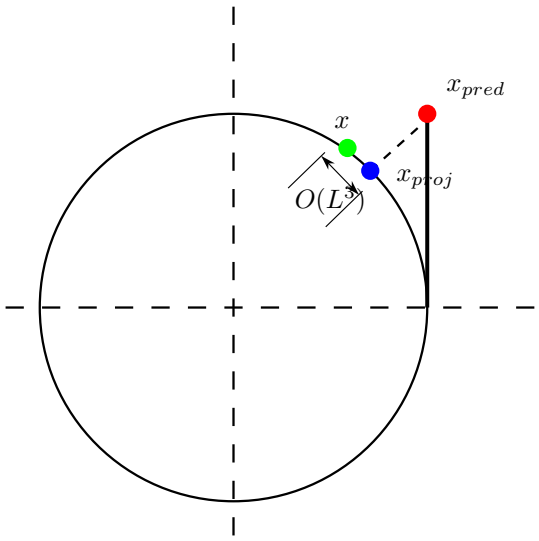

Figure 8: *The red dot is the predicted position, the blue dot is the projected position and the green dot is the correct position.*

decay[1] and thus also allows a stiff limit. Assuming constant step sizes, BDF2 can be written as:

$$\begin{aligned} x^{(n+1)} &= \tfrac{4}{3}x^{(n)} - \tfrac{1}{3}x^{(n-1)} + \tfrac{2}{3}\Delta t v^{(n+1)} \\ v^{(n+1)} &= \tfrac{4}{3}v^{(n)} - \tfrac{1}{3}v^{(n-1)} + \tfrac{2}{3}\Delta t M^{-1}F(x^{(n+1)}) \end{aligned} \tag{20}$$

Eliminating velocity as before gives:

$$\begin{aligned} x^{(n+1)} = \tfrac{4}{3}x^{(n)} - \tfrac{1}{3}x^{(n-1)} + \tfrac{8}{9}\Delta t v^{(n)} - \tfrac{2}{9}v^{(n-1)} \\ + \tfrac{4}{9}\Delta t^2 M^{-1}F(x^{(n+1)}) \end{aligned} \tag{21}$$

Breaking this up into a prediction step followed by a first order projection to the constraint:

$$\begin{aligned} x_0 &= \tfrac{4}{3}x^{(n)} - \tfrac{1}{3}x^{(n-1)} + \tfrac{8}{9}\Delta t v^{(n)} - \tfrac{2}{9}v^{(n-1)} \\ &\quad + \tfrac{4}{9}\Delta t^2 M^{-1}F \\ x^{(n+1)} &= \text{project}(x_0) \end{aligned} \tag{22}$$

Finally the time $n+1$ velocity is taken from the BDF2 formula for the position update, which can be rewritten as:

$$v^{(n+1)} = \tfrac{1}{\Delta t}\left[\tfrac{3}{2}x^{(n+1)} - 2x^{(n)} + \tfrac{1}{2}x^{(n-1)}\right] \tag{23}$$

The basic steps of the algorithm are the same as before, just with a few extra vector adds in finding the predicted position and final velocity, which is

of negligible cost compared to the projection operation. The error analysis is nontrivial, since at first glance the second order truncation error in the projection would seem to give a globally first order method. However, while the predicted position is indeed $O(\Delta t^2)$ away from the constraint manifold, most of this error is normal to the manifold itself and is eliminated by projection: numerical experiments indicate that after projection the position only suffers $O(\Delta t^3)$ truncation error, and together with the third order error in velocity, this results in a globally second order accurate algorithm. This can easily be verified for the simple case of motion constrained to a circle as shown in figure 8, for example: a step of length L off the circle on a tangent gives a predicted position $x_{pred} = (1, L)$. The correct position is $x = (cos(L), sin(l))$, giving a distance error of $|x_{pred} - x| = |(1 - cos(L), L - sin(L))|$ which when cos and sin are represented with Taylor series gives,

$$\begin{aligned} |x_{pred} - x| &= |(1 - cos(L), L - sin(L))| \\ &= |(1 - (1 - O(L^2)), L - (L + O(L^3)))| \\ &= |(O(L^2), O(l^3))| \\ &= \sqrt{O(L^2)^2 + O(L^3)^2} \\ &<= O(L^2) \end{aligned} \tag{24}$$

Instead, the predicted position projected back to the cicle is $x_{proj} = \frac{x_{pred}}{|x_{pred}|} = (\frac{1}{\sqrt{1+L^2}}, \frac{L}{\sqrt{1+L^2}})$. Using a Taylor series to represent both terms in the distance error here results in a cancellation,

$$\begin{aligned} |x_{proj} - x| &= (\frac{1}{\sqrt{1+L^2}} - cos(L), \frac{L}{\sqrt{1+L^2}} - sin(L)) \\ &= ((1 - \frac{L^2}{2} + O(L^4)) - (1 - \frac{L^2}{2} + O(L^4)), (L + O(L^3)) - (L + O(L^3))) \\ &= (O(L^4), O(L^3)) \\ &= \sqrt{O(L^4)^2 + O(L^3)^2} \\ &<= O(L^3) \end{aligned} \tag{25}$$

Additional benefits include an order of magnitude less numerical dissipation, due to higher accuracy, and typically improved projection times: by using additional information from previous steps about the constraint manifold, the predicted position tends to stay closer to the manifold and requires fewer iterations in projection.

5.2 Constraint Projection

Given the constraint as defined in (1), I take the first order Taylor series

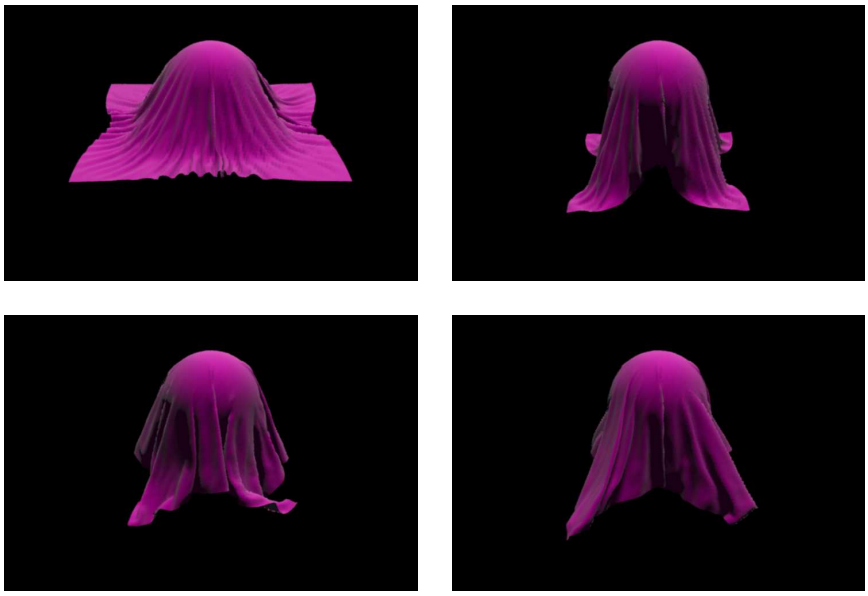

Figure 9: *A developable surface is dropped on a sphere, with immediate wrinkling and creasing patterns.*

$$\begin{aligned} C_{x_0}(x) &= C(x_0) + \frac{\partial C(x_0)}{\partial x}(x - x_0) + O((x - x_0)^2) \\ C(x) &= C + J(x - x_0) + O((x - x_0)^2) \end{aligned} \tag{26}$$

into which I substitute the the lagrange multiplier force, as defined in (2), to get out updated values

$$\begin{aligned} 0 &= C + J(x_0 + W^{-1}J^T\lambda - x_0) \\ &= C + JW^{-1}J^T\lambda \end{aligned} \tag{27}$$

I then solve for the lagrange multiplier,

$$\lambda = -(JW^{-}1J^T)^{-1}C + O((x - x_0)^2) \tag{28}$$

and substitute this back into the updated formula for the position update, giving the final postion

$$x = x_0 - W^{-1}J^T(JW^TJ)^{-1}C \tag{29}$$

This gives a second order approximation of x_0 projected onto the constraint manifold. In order to make this algorithm more robust, I iteratively update the Taylor series terms. As a final improvement I perform a line search at each iteration starting with $l = 1$, halving it until the relative error of the constraint is less than that of the previous iteration. The update formula is given,

Figure 10: *Collisions are robustly handled as a nondynamically driven figure walks wearing a cape.*

$$x_{i+1}(l) = x_i - l(W^{-1}J(x_i)^T(J(x_i)W^TJ(x_i))^{-1}C(x_i)) \tag{30}$$

The projection stage is completed when $C(x_i)/d$ is below some tolerance, where d is the edge lengths as defined in (1).

6 Results

Figures 1, 3 and 9 show example frames from simulations. I rendered the raw conforming mesh, with smoothed vertex normals but no subdivision. The meshes were 100×100, and simulations ran at 9.52 seconds/step with time steps of 1ms on an Athlon 64 3500+, with PARDISO [20] as the linear solver. I ran fast projection with a tolerance on maximum relative error of 10^{-4}, taking 10 Newton steps on average. Due to step size, collision handling was a small fraction of total run time.

I also simulated a moving skinned character wearing a cape (see figure 10) to evaluate the new integration scheme; simulation time was reduced to 3.96 from 5.37 seconds/step for the single step scheme, as the multistep method required on average half the number of iterations for the projection stage to converge. Similarly a skinned character wearing an irregularly meshed shirt (see figure 3), of higher resolution than the cape, ran at 4.58 seconds/step using the new

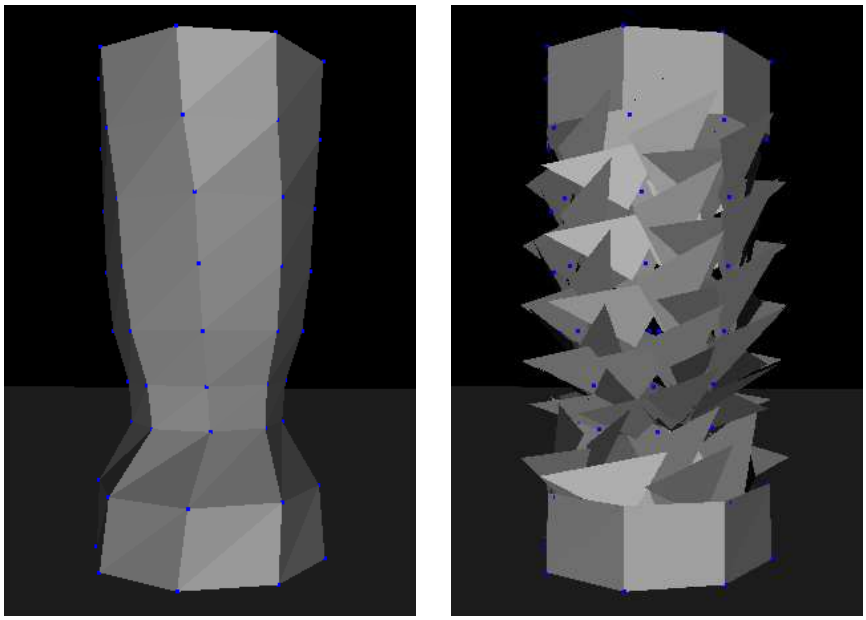

Figure 11: *The coupled ghost mesh collapses as the conforming mesh folds in upon itself. The rest state is that of a perfectly regular cylinder.*

scheme.

7 Conclusions

I have presented an effective new discretization for deformable surfaces which can robustly handle developable surfaces without locking artifacts at any mesh resolution. Since the underlying nonconforming simulation mesh isn't continuous, I couple in a conforming ghost mesh to handle contact and collisions and for rendering. In addition, I provided a second order accurate multistep constrained mechanics time integration scheme based on BDF2, using just position projection, which both accelerates fast projection and significantly reduces numerical damping.

8 Future Work

In future work I am particularly interested in resolving the issues brought up by the ghost conforming mesh, such the collapsing cylinder show in figure 11 as well as rigorously analyzing the momentum transfer and energy dissipation of the coupling method. Edge subdivision schemes (in lieu of simple vertex averaging) may improve the smoothness of the rendered output, and coupling

internal dynamics and collisions, similar to Baraff and Witkin's method[2], may avoid perturbations from developability caused by time splitting.

Acknowledgements

This work was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada.

References

[1] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, Philadelphia, PA, USA, 1998.

[2] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proc. ACM SIGGRAPH*, pages 43–54, 1998.

[3] M. Bergou, M. Wardetzky, D. Harmon, D. Zorin, and E. Grinspun. A quadratic bending model for inextensible surfaces. In *Symp. Geometry Processing*, pages 227–230, 2006.

[4] Pengbo Bo and Wenping Wang. Geodesic-controlled developable surfaces for modeling paper bending. *Computer Graphics Forum*, 26(3):365–374, 2007.

[5] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods (2nd ed.)*. Springer, 2002.

[6] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 21(3):594–603, 2002.

[7] Robert Bridson, Sebastian Marino, and Ronald Fedkiw. Simulation of clothing with folds and wrinkles. In *Symp. Comp. Anim.*, 2003.

[8] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 21(3):604–611, 2002.

[9] Richard S. Falk. Nonconforming finite element methods for the equations of linear elasticity. *Math. Comp.*, 57(196):529–550, 1991.

[10] Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. Efficient simulation of inextensible cloth. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 26(3), 2007.

[11] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Computer Graphics*, 30(Annual Conference Series):171–180, 1996.

[12] Michael Hauth. *Visual Simulation of Deformable Models*. PhD thesis, July 2004.

[13] Geoffrey Irving, Craig Schroeder, and Ronald Fedkiw. Volume conserving finite element simulations of deformable models. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 26(3):13, 2007.

[14] Y. M. Kang and H. G. Cho. Bilayered approximate integration for rapid and plausible animation of virtual cloth with realistic wrinkles. In *Computer Animation, IEEE Computer Society*, pages 203–214, 2002.

[15] Yannick L. Kergosien, Hironobu Gotoda, and Tosiyasu L. Kunii. Bending and creasing virtual paper. *IEEE Comput. Graph. Appl.*, 14(1):40–48, 1994.

[16] Yong-Jin Liu, Kai Tang, and Ajay Joneja. Modeling dynamic developable meshes by the hamilton principle. *Comput. Aided Des.*, 39(9):719–731, 2007.

[17] Jörg Peters and Ulrich Reif. The simplest subdivision scheme for smoothing polyhedra. *ACM Trans. Gr.*, 16(4):420–431, 1997.

[18] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface '95*, pages 147–154, 1995.

[19] X. Provot. Collision and self-collision handling in cloth model dedicated to design garment. *Graphics Interface*, pages 177–89, 1997.

[20] O. Schenk and K. Gärtner. On fast factorization pivoting methods for symmetric indefinite systems, 2006.

[21] K. D. Tsiknis. Better cloth through unbiased strain limiting and physics-aware subdivision. Master's thesis, University of British Columbia, 2006.

[22] P. Volino and N. Magnenat-Thalmann. Fast geometric wrinkles on animated surfaces. In *7th Intl. Conf. in Central Europe on Computer Graphics and Visualization (WSCG)*, 1999.

[23] M. Wardetzky, M. Bergou, D. Harmon, D. Zorin, and E. Grinspun. Discrete quadratic bending energies. *Computer Aided Geometric Design*, 2007.