# Making Space for Cloth Simulations Using Energy Minimization

David Minor
Digital Domain 3.0
dminor@d2.com

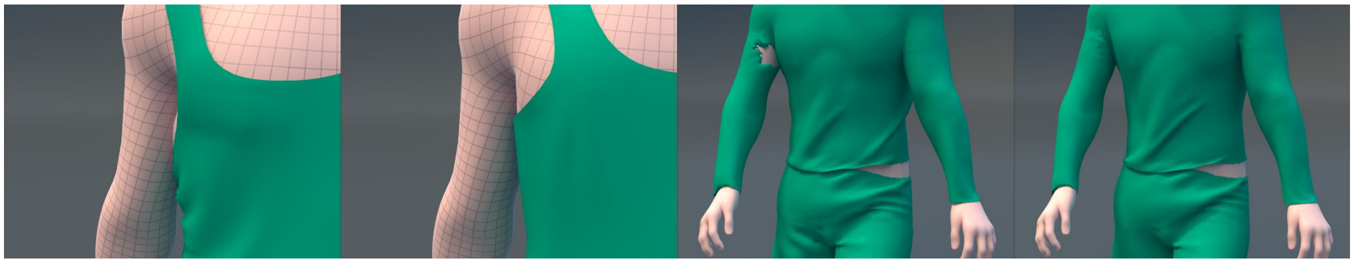

**Figure 1: Using our method to correct unwanted bunching and jittering by adding a gap in the armpit area in two scenarios**

## ABSTRACT

Geometry interpenetrations are a common issue in creature effects workflows, particularly in cases which require simulations, for example hair and clothing. Production rigs often introduce self intersections in regions like armpits and elbows, which can cause ugly instabilities and undesirable behavior when they interfere with simulated objects such as garments. To achieve visually acceptable results, these simulations often require a small gap to allow sliding between opposing surfaces, and the process of making these modifications can often be quite manual. Here we present a production proven creature effects tool for resolving these issues automatically.

## CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**; **Collision detection**;

## KEYWORDS

ADMM, Projective Dynamics, Gradient Deformation Transfer

**ACM Reference Format:**
David Minor. 2018. Making Space for Cloth Simulations Using Energy Minimization. In *Proceedings of SIGGRAPH '18 Talks*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3214745.3214764

## 1 OVERVIEW

Untangling Cloth [Baraff et al. 2003] addresses problems with self-intersection in the simulator itself by identifying trapped areas and pinning them in place in a process called flypapering. It is often desirable to add a gap between the actual surfaces though, for example to prevent crumpling in a cloth simulation, which can be visually distracting, even when physically correct. We also wish to be able to use commercial simulation software packages, whose implementation we do not have access to.

The input to our procedure is a triangle mesh which may have regions that self-intersect or come too close to an opposing surface. The output should be a mesh with intersections removed and appropriate margins introduced. We will not try to enforce this strictly, but we present a method that works well enough to be practically useful.

The desired outcome is that the *self-interfering* portions of the mesh, regions that self-intersect or are too close to an opposing surface, are moved such that they no longer interfere. These modifications need to be smoothed out non-locally though, or we will get ugly looking discontinuities.

One way of doing this is by treating the surface as a physical object with a resistance to deformation and finding a balance between repulsive forces pushing opposing surfaces away from each other and forces maintaining the shape and position of the surface. This can also be viewed as an energy minimization. If we write an energy function that takes on high values when self-interference is present in the surface, but also has high values when the surface has deformed severely relative to its original configuration, we can use a standard method to minimize this energy function and retrieve a smooth surface with interpenetrations removed.

## 2 ENERGIES AND FORCES

The energy we minimize has three terms, each of which penalize different things. Prior to discretization on a triangle mesh, we can write it as an area integral over the surface $\Omega$:

$$E = \frac{1}{2}\int_{\Omega}(k_d|\boldsymbol{F} - \boldsymbol{F_0}|^2 + k_a|\boldsymbol{x} - \boldsymbol{x}_0|^2 + k_i d^2)dA \qquad (1)$$

(1) $k_d|\boldsymbol{F} - \boldsymbol{F_0}|^2$: This deformation term discourages bending, shearing and stretching, and tries to force the deformation gradient, $\boldsymbol{F}$ towards towards its original value, $\boldsymbol{F_0}$, similar to the deformation energy used in [Sumner and Popović 2004]. In the artist tool we fix $k_d$ to 1.

(2) $k_a|\boldsymbol{x} - \boldsymbol{x}_0|^2$: This is an anchoring term, which discourages overall motion of the vertices. We allow the user to paint

$k_a$ as a map, as they may want to vary the mobility or pin vertices kinematically. A typical value is 0.5.

(3) $k_i d^2$: This interference term tries to force interpenetrations apart and create a gap between opposing surfaces. The variable $d(\boldsymbol{x})$ is a penetration depth which increases from zero as parts of the surface approach each other within the collision margin and start to intersect. A typical value for $k_i$ is around 10.

## 3 COLLISION FORCES AND DETECTION

### 3.1 Self-Intersection

We can detect if a general point is inside a closed mesh by measuring the winding number at that point, i.e. the number of back facing triangles minus the number of forward facing triangles encountered by a ray travelling in an arbitrary direction from the point. If the winding number is zero, the point is outside the mesh.

We wish to test the winding number at a vertex. However, the winding number has two possible values on the surface depending on which side you measure it on. We choose to measure the winding number immediately outside the surface at the vertex by shooting a ray in an outward facing direction and excluding all triangles connected to the vertex from the ray test. We use angle weighted pseudo normals to calculate suitable outward facing directions at each vertex [Baerentzen and Aanaes 2005].

If the winding number is non-zero outside the vertex it is self-intersecting and we measure the penetration depth by tracing backward along the pseudo normal to the first forward facing hit, measuring the length of the ray and adding the margin distance.

*3.1.1 Proximity Collisions.* We also need to detect collisions if the mesh is not self-intersecting but the gap is too small. We do this by attaching spheres (with a diameter equal to the margin) to each vertex, centered half a margin away from the vertex in the direction of the pseudo normal. We then do an intersection test between these spheres and the surface, using the nearest point on the triangle furthest along the normal as the collision point. The penetration depth is the collision margin minus the distance to this point.

### 3.2 Energy Minimization

We discretize equation 1 on a triangle mesh and minimize the energy using ADMM [Overby et al. 2017], without the terms the authors use to simulate dynamics. We handle the deformation energy term using the proximal operator method the authors describe, but we include the collisions and the anchor springs in the global minimization phase. Because the collisions introduce terms in this minimization that vary from iteration to iteration, we use a Parallel Gauss-Seidel solver [Fratarcangeli et al. 2016] instead of a Cholesky factorization, as this can be rapidly reconfigured for the varying self-collision terms and converges quickly for triangle mesh simulations. We run the method for a small fixed number of ADMM iterations, normally four are sufficient for acceptable results.

## 4 RESULTS

The tool is usually used on restricted portions of the mesh, and the performance depends on the complexity of the geometry, but 0.1s to 0.2s are typical processing times for a 30000 vertex mesh with 5000 active vertices.

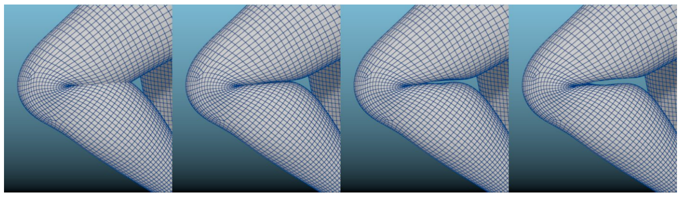

**Figure 2: Using our method to resolve interpenetrations and add various different margins in a knee area**

Figure 1 shows a clothing sim with and without our procedure applied to the collision geometry. In the vest sim on the left hand side, intersections in the armpit area have pushed the cloth down in an unconvincing, visually distracting way. Processing the geometry with our tool adds a gap in the armpit area, preventing this behavior. On the right hand side we simulate a figure wearing a shirt and shorts using Maya nCloth. Interpenetrations in the armpit area have caused instability in the cloth sim, which our method fixes.

Figure 2 Shows our method applied to a knee area, with the original geometry shown on the left and the margin distance set to 0, 0.5 and 1cm.

## ACKNOWLEDGMENTS

I would like to thank Matthew Overby for implementing the parallel Gauss-Seidel solver, Jimmy Gordon for helping me track down demonstration materials, Jason Iverson for his renders and Doug Roble, Nafees Bin Zafar and Gene Lin for their feedback.

## REFERENCES

J. Andreas Baerentzen and Henrik Aanaes. 2005. Signed Distance Computation Using the Angle Weighted Pseudonormal. *IEEE Transactions on Visualization and Computer Graphics* 11, 3 (May 2005), 243–253. https://doi.org/10.1109/TVCG.2005.49

David Baraff, Andrew Witkin, and Michael Kass. 2003. Untangling Cloth. *ACM Trans. Graph.* 22, 3 (July 2003), 862–870. https://doi.org/10.1145/882262.882357

Marco Fratarcangeli, Valentina Tibaldo, and Fabio Pellacini. 2016. Vivace: a practical gauss-seidel method for stable soft body dynamics. *ACM Trans. Graph.* 35, 6 (2016), 214:1–214:9. http://dl.acm.org/citation.cfm?id=2982437

Matthew Overby, George E. Brown, Jie Li, and Rahul Narain. 2017. ADMM / Projective Dynamics: Fast Simulation of Hyperelastic Models with Dynamic Constraints. *IEEE Transactions on Visualization and Computer Graphics* 23, 10 (10 2017), 2222–2234. https://doi.org/10.1109/TVCG.2017.2730875

Robert W. Sumner and Jovan Popović. 2004. Deformation Transfer for Triangle Meshes. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 399–405. https://doi.org/10.1145/1015706.1015736