Untangling Cloth With Boundaries

Martin Wicke Hermes Lanker Markus Gross

Computer Graphics Laboratory, ETH Zurich

Abstract

This paper presents a history-free collision handling algorithm for cloth. Without information about past timesteps, our method resolves intersections on cloth with boundaries. Using a global intersection analysis, we determine interpenetrating regions of the cloth. We analyze the possible intersection paths on 2D manifolds with boundaries and present a classification of intersection paths according to the position of their endpoints and the number of degenerated vertices on the path.

Radial basis function (RBF) fitting is used to extrapolate correspondence information available on the intersection paths to the surrounding cloth. Attractive forces between corresponding points in the interpenetrating cloth region resolve the intersections. Our method can be applied as a preprocessing step to ensure that modeling results are suitable for animation. It can also be used during animation to correct intersections introduced by constraints.

1 Introduction

Collision handling for cloth is a notoriously hard problem in computer graphics. Due to the two-dimensional nature of cloth, collision handling methods for solids that typically rely on penetration depth are not applicable.

A viable alternative to these approaches are algorithms that prevent intersections based on continuous collision detection. Without external constraints, these methods can garantuee that if the last timestep was intersection-free, the next timestep is also interection-free. Continuous collision detection identifies all primitives that crossed parts of the cloth, such that those crossovers can be prevented.

Unfortunately, tangling during simulation is not a purely academic possibility, but routinely happens during animation. If the cloth simulation is subject to external constraints such as collisions with solid objects, these constraints might force the cloth into an illegal state even in the presence of collision handling.

Intersection prevention methods cannot recover from these situations. If some part of the cloth ends up on the wrong side, the collision handling algorithm will prevent it from crossing over to the correct side.

All methods relying on a legal prior state fail if such a state is not available. The *history-free* method proposed in [2] works around this problem by globally analyzing the current (potentially illegal) state of a piece of cloth, and proposing actions to resolve intersections based on the current state of the simulation alone.

History-free methods solve some of the problems in collision detection for cloth, but they cannot replace traditional collision detection. So far, the available methods have not been general enough to present an alternative to history-based collision detection. Also, due to the inherently global nature of the problem, history-free collision detection methods are slow compared to traditional approaches.

1.1 Contributions

Similar to [2], we use *global intersection analysis* (GIA) to determine which regions of the cloth are on the wrong side. We extend their path finding and classification method to handle situations involving boundaries. We analyze the possible intersections and present a classification of all possible intersection paths considering boundaries.

We propose a method for finding point-to-point correspondences between intersecting parts of the cloth. Using radial basis function (RBF) fitting in parameterization space, we extrapolate the correspondence information available on the intersection path to its surroundings. This technique enables robust collision response whenever parts of the cloth are recognized to be on the wrong side.

When boundaries are considered, not all possible intersections result in parts of the cloth being

clearly on the wrong side. We propose methods for collision response to such intersections.

The remainder of this paper is organized as follows: We discuss relevant prior research in Section 2. After giving an overview of the proposed method in Section 3, we describe our intersection analysis and present a classification of possible intersection paths in Section 4. Section 5 presents our methods for collision handling. In Section 6, we show results. Finally, we discuss strengths and weaknesses and point out some directions for future research in Section 7.

2 Related Work

Starting with the pioneering work of Terzopoulos et al. [13], various approaches have been developed for computing the dynamic behaviour of two-dimensional elastic objects [6, 12, 1, 9, 7, 8, 4, 15].

However, collisions between sheets of cloth have proven to be more difficult to handle. Simple collision handling approaches imposing penalty forces on vertices that lie inside objects are feasible for cloth-object collisions but, due to the infinitesimal thickness of cloth, they cannot be applied to cloth-cloth collisions. A solution to this problem is continuous collision detection. Bridson et al. [3] avoid crossovers entirely and guarantee that the cloth is free of intersections at the end of a timestep if it was free of intersections before.

The earlier work of Volino et al. [16, 17] first allows collisions, and later corrects collisions based on local criteria. They use a statistical approach to determine which parts of the cloth need to be corrected.

These approaches work well for unconstrained cloth, however, the collision detection and response has to be integrated into the simulation not only of the cloth, but also of other objects in the scene. If other objects do not behave physically, for example because they are animated using motion capture data, cloth might be forced into a tangled state despite the efforts of collision handling schemes.

Baraff et al. [2] propose a history-free intersection handling method to remedy these issues. However, they only treat intersections possible on meshes without boundaries. Although some of their results can be applied to meshes with boundaries, they state that their method is not general enough to "untangle boundary intersections".

Recently, Volino and Magnenat-Thalmann [18] proposed a method for resolving intersections based on minimization of the intersection paths. Their algorithm does not require a full GIA to resolve small intersections, and it works for most cases involving boundaries.

This paper generalizes the results of [2] to meshes with boundaries and proposes strategies to untangle cloth with boundaries. Our method makes no assumptions on the current state of the cloth. It can be integrated into any existing cloth animation framework.

3 Algorithm Outline

Our algorithm uses global intersection analysis to determine which parts of the cloth have crossed over to the wrong side. This analysis computes all intersections between cloth primitives, and assembles them into intersection paths. In our discussion, we will assume that the cloth is represented by a triangle mesh. The results are equally valid for other representations such as quadrangle meshes or spline patches. After the intersection curves have been computed (see for example [11]), the intersection handling can be applied unmodified.

For simplicity, only the case of self-intersections is treated in this paper, however, our results generalize trivially to intersections between several pieces of cloth.

We assume that the cloth is parameterized in some two-dimensional space $\mathbb{U}$. The parameterization needs to be smooth, but the parameter space $\mathbb{U}$ is arbitrary. Thus, a scarf might best be parameterized on a square, while a skirt can be parameterized on a cylinder. Conformal or area-preserving parameterization is not necessary, but improves the results of the correspondence computation (see Section 5.1).

In world space, each intersection path is a single curve. Since each part of the intersection path is formed by the intersection of two triangles, each point on the intersection path touches two points on the cloth (except for *loop vertices*, see below). Thus, each point on the intersection path has two parameterization values. A single intersection creates two paths in the parameter domain, where most of the intersection analysis takes place. Hence, paths are arranged in pairs reflecting a common world space position. See Figure 1 for examples. After the inter-

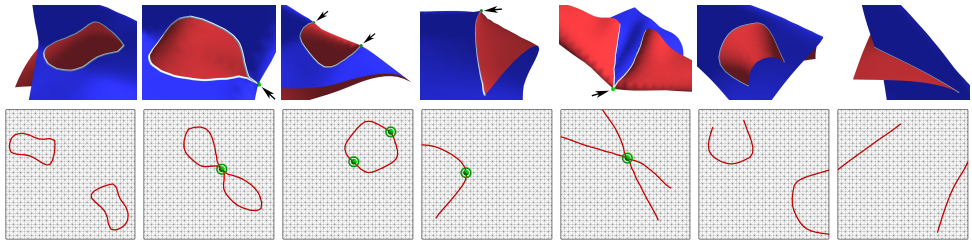


Figure 1: Different paths in world space (top row) and parameter domain $\mathbb{U}$ (bottom row). Loop vertices are marked on the paths. (a) a pair of CLOSED paths. (b) EIGHT path. (c) loop-loop (LL) path. (d) boundary-loop-inside (BLI) path. (e) CROSS path. (f) boundary-boundary/inside-inside (BB/II) path pair. (g) boundary-inside (BI) path pair.

section paths are known, they are classified according to intrinsic properties.

For each path separately, we partition the cloth into regions that are *inside*, i. e. regions that have crossed over and for which response is necessary, and *outside* regions. Figure 2 shows inside regions for different path types. Some paths involving a boundary do not define an inside region.

Finally, intersection handling is performed on each path. The goal in intersection handling is to minimize the *inside* region of the path. To this end, we compute target points for mesh vertices that are *inside*. Moving the mesh vertices towards their target points resolves the intersection. For paths that do not have an inside region, we push the intersection path to the nearest cloth boundary.

4 Global Intersection Analysis

We will first summarize the GIA before turning our attention to the analysis of possible path types. As a first step of the GIA, intersections between cloth primitives (triangles) are computed. We use spatial hashing [14] to speed up intersection computation.

Each triangle/triangle intersection yields two *corresponding* intersection *segments*. These two segments share a common position in world space, but each segment has a different position on the cloth, corresponding to one of the intersecting triangles.

The set of intersection segments is assembled into *intersection paths*. Baraff et al. [2] describe how to trace the intersection paths given the intersection segments.

Paths that consist of corresponding segments are called corresponding paths. Note that for some path types, a path can be its own corresponding path. Figure 1 shows intersections and the resulting paths in the parameter domain $\mathbb{U}$.

One case deserves special attention. If two intersecting triangles share a common vertex, this vertex is called a *loop vertex*. Intersection paths that encounter a loop vertex reverse direction. The number of loop vertices on a path is an important criterion for path classification. A path with a loop vertex is its own corresponding path. Intersection paths fold back onto themselves in world space after reaching a loop vertex (e. g. Figure 1 (d)). Since it is possible that several paths meet in a single loop vertex, care has to be taken to follow the correct path upon reaching a loop vertex.

4.1 Path Types

We found it useful to analyze the intersection paths in the parameter space $\mathbb{U}$. Criteria for path classification are whether the path is closed or where it ends, as well as the number of loop vertices on the path. Figure 1 shows the possible path types.

In the parameterization domain, a path can either end on a boundary or somewhere inside, or the path is closed. On 2D manifolds, a path can reverse direction at most twice before forming a loop, hence there cannot be more than two loop vertices in a single path. The list below discusses all possible path types.

CLOSED paths are the simplest intersection paths. A CLOSED path contains no loop vertices. It forms a loop in world space as well as in parameter space. The corresponding path to a CLOSED path is a CLOSED path.

EIGHT paths are closed paths with a single loop vertex. This rare path occurs when two CLOSED paths share exactly one common vertex, which becomes the loop vertex. EIGHT paths are a single loop in world space, while they unfold to an 8-shaped path in the parameterization domain $\mathbb{U}$. If we encounter such a path, we split it at its loop vertex, creating two CLOSED paths.

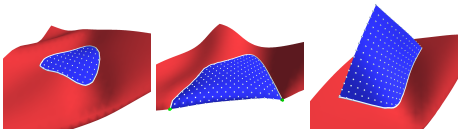
Loop-Loop (LL) paths are closed and contain two loop vertices. LL paths form a loop in the parameter domain, but fold into a single line in world space. In our framework, these paths can can be treated exactly like CLOSED paths.

If the path is not closed, but has a loop vertex, then at least one end of the path lies on the boundary. We call this a boundary-loop-inside (BLI) path. The case where both end points lie on the boundary (boundary-loop-boundary, BLB path) is extremely rare, and does not invalidate any of the discussion concerning BLI paths. When we refer to BLI paths below, we implicitly include BLB paths. A BLI path can be thought of as half a LL path. BLI paths are more difficult to handle than the closed path varieties, since they to not partition the cloth mesh into separate components.

There is the theoretical possibility of an open path with two loop vertices. The path starts and ends on a boundary, and the boundary vertices touch. Following our naming convention, this path is called boundary-loop-loop-boundary (BLLB) path. The configuration leading to this path type is unstable, and the path will split into two BLI paths as soon as the boundary vertices assume different positions. BLLB paths can be split into two BLI paths for handling.

There is an open version of the EIGHT path, which we will call CROSS path. It can also be seen as two BLI paths sharing a common loop vertex. Similar to the EIGHT path, we handle this path type by splitting it at the loop vertex, yielding a BB/II path pair (see below).

If there are no loop vertices on an open path, three possible path types remain. The end points of a path can either lie on the boundary or inside the cloth. If both endpoints lie on the boundary, we call the path a boundary-boundary (BB) path. Similarly, if only one or no endpoint lies on the cloth boundary, it is called boundary-inside (BI) or inside-inside (II) path respectively. The corresponding path to an II path is always a BB path, while a BB path can also correspond to a BI path. The most common, however, are BB/II and BI/BI pairs. Of these open path types, only BB paths partition the cloth mesh, thus BI/BI path pairs are difficult to handle. We address the problem in Section 5.3. BB/II and BB/BI pairs can be handled in the same way. We will only explicitly discuss the more common BB/II pair.



(a) (b) (c)

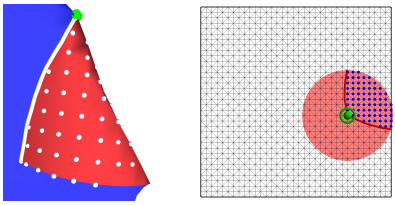
Figure 2: Inside regions (blue) and vertices of a (a) CLOSED (b) LL (c) BB path are highlighted. Parallel flood fill on both sides of the path is used to determine the smaller part of the cloth, which is then labelled as inside.

4.2 Determining Inide/Outside Regions

Once the path type is known, we can determine *inside* and *outside* regions on the cloth. For each intersection path individually, we decide which part of the cloth is on the wrong side. The *inside* regions need to be corrected.

The CLOSED, LL, and BB paths are *partitioning* paths, i. e. intersection paths that partition the mesh into two components. Figure 2 shows inside regions of these paths. Like [2], we use parallel flood-fill to determine which of the two components is smaller. That region is then labelled as *inside* with respect to the path. This procedure is consistent with the goal of using the smallest possible correction to resolve the intersections. It is also a sensible choice if used during a simulation. During a short period of time (since the last untangling step), it is more likely that small penetrations have occured.

The situation is more complicated for paths that are not partitioning, i. e. BLI, BI, and II paths. For BLI paths, we use a heuristic to determine an *inside* region. As can be seen in Figure 3 (a), the cloth around a BLI path forms a roughly conical shape with the loop vertex in its apex. Note that if the cone is flattened, the intersection is resolved. In order to define an *inside* region for BLI paths, we restrict our attention to a circular patch of cloth around the loop vertex, just large enough to contain at least one of the end points of the BLI path. We then use the parallel flood-fill algorithm as described above, re-



(a) (b)

Figure 3: (a) Cloth configuration around a BLI path. The path is a seamline of a cone with the loop vertex at its apex. Cloth vertices in the *inside* region of the path are highlighted. (b) *Inside* region in the parameter domain $\mathbb{U}$. The circular region of interest is highlighted, as well as the *inside* cloth vertices.

stricted to our region of interest. Figure 3 shows the result of this method.

BI paths require special care. Although it would be possible to assign *inside* regions for BI paths using heuristics, we have found these methods to be too unreliable in a general setting. Section 5.3 describes a method for dealing with these paths.

No *inside* region is defined for II paths. However, since the corresponding path to an II path is always a BB path, we can use the *inside* region of the corresponding BB path for intersection handling.

5 Intersection Handling

Intersection handling is based on the *inside* regions defined during the GIA. The idea is to find a target point $\mathbf{r}_i'$ for each cloth vertex $\mathbf{r}_i$ in the *inside* region, such that if all $\mathbf{r}_i$ are moved towards the respective $\mathbf{r}_i'$, the intersection is resolved.

Baraff et al. [2] used the closest point in the *inside* region of the corresponding path as target point for a vertex. This method only works for CLOSED path pairs, since this is the only path type defining two separate *inside* regions.

In constrast, our method works for all path pairs where at least one path defines an *inside* region, i. e. all path types except BI paths (BI paths are treated in Section 5.3). Once the target points are known, we can either apply penalty forces to resolve the intersection or directly displace the cloth vertices.

5.1 Finding Target Points

Vertices in the *inside* regions of cloth have crossed over to the wrong side. If we knew where exactly they had penetrated the cloth, correcting the intersections would be easy. Vertices could simply be moved back to the point of initial contact. Approaches using continuous collision detection use this idea to resolve collisions.

Information about the vertices' history is not available to us. However, the correspondences between the intersection paths contain the information we need. We extrapolate this information to the *inside* region of the path.

Consider a parameterized surface $\mathbf{s}(\mathbf{u}), \mathbf{u} \in \mathbb{U}$ and intersection paths $P = \{\mathbf{u}(t)\} \subset \mathbb{U}$, and its corresponding path $P' = \{\mathbf{v}(t)\} \subset \mathbb{U}$, both parameterized in $t \in [0,1]$ such that $\mathbf{u}(t)$ and $\mathbf{v}(t)$ are corresponding, i. e. $\mathbf{s}(\mathbf{u}(t)) = \mathbf{s}(\mathbf{v}(t))$.

We are looking for a smooth function $\mathbf{d}_P : \mathbb{U} \to \mathbb{U}$, mapping each point $\mathbf{r}$ in the *inside* region to its corresponding target point $\mathbf{r}' = \mathbf{d}_P(\mathbf{r})$. We know the values of $\mathbf{d}_P$ for all points on the path:

$$\forall t \quad \mathbf{d}_P(\mathbf{u}(t)) = \mathbf{v}(t). \tag{1}$$

We use RBF fitting to construct such a smooth function $\mathbf{d}_P$. For an introduction to the topic, we refer the reader to [5]. RBF interpolation functions are of the form:

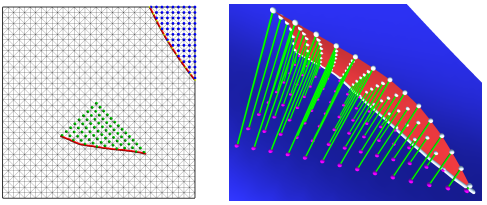
$$d(\mathbf{x}) = p(\mathbf{x}) + \sum_j w_j \phi(\|\mathbf{x} - \mathbf{c}_j\|), \tag{2}$$

where w_j are weights, $\mathbf{c}_j$ are the *centers* of the RBF fit, ϕ is the radial basis function, and $p(\mathbf{x}) = \sum_j a_j p_j(\mathbf{x})$ is a polynomial. $\{p_j\}$ is the set of monomials. In our implementation, we use $\phi(x) = x^2 \log x$ and a bivariate polynomial of degree one.

For the RBF fit, we place equidistant samples on the path P. These n samples $\mathbf{u}_i = \mathbf{u}(t_i)$ are the centers of the RBF. Their number n is chosen according to the path length. Typical values of n range from 5 to 30 centers. To obtain the weights w_j and polynomial coefficients a_j, we solve the system of equations given by $d(\mathbf{u}_i) = \mathbf{v}_i$, where $\mathbf{v}_i = \mathbf{v}(t_i)$ is the point on P' corresponding to $\mathbf{u}_i$.

$$\begin{pmatrix} \Phi & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} w_j \\ a_j \end{pmatrix} = \begin{pmatrix} v_i \\ 0 \end{pmatrix} \tag{3}$$

Here, $\Phi_{ij} = \phi(\|\mathbf{u}_i - \mathbf{u}_j\|)$ and $A_{ij} = p_j(\mathbf{u}_i)$ (with a polynomial of degree one, $A \in \mathbb{R}^{n \times 4}$). Note that



(a) (b)

Figure 4: (a) A path pair consisting of a BB path P (top-right) and an II path P' (middle). Shown are the inside vertices r_i of P as well as their corresponding points $\mathbf{d}_P(r_i)$. (b) *Inside* vertices of P and their target points in world space. Attractive forces between corresponding points resolve the intersection.

the matrix A is singular if the centers $\mathbf{u}_i$ lie on a straight line [10]. Adding random noise to the $\mathbf{u}_i$ solves the problem. We never use less than three centers.

The function $\mathbf{d}_P$ is two-dimensional and we solve for weights for its components separately. These weight can be used in (2) to construct a correspondence function $\mathbf{d}_P = (d_1, d_2)$. Figure 4 illustrates the fitting process.

5.2 Applying Corrections

In order to resolve the collisions, we apply a penalty force to all vertices in the *inside* regions such that the interpenetrating parts of the cloth are separated. This force points in the direction of the target point.

$$\mathbf{F}(\mathbf{r}) = \frac{\mathbf{r}' - \mathbf{r}}{\|\mathbf{r}' - \mathbf{r}\|} \max(k\|\mathbf{r}' - \mathbf{r}\|, f_{\max}) \quad (4)$$

We use a force that linearly increases with the distance to the corresponding point (the equivalent of penetration depth), up to a maximum value. Note that in a physical simulation, the correction forces should be applied symmetrically to r and r' to ensure conservation of linear and angular momentum.

For the untangling process to be successful, it is necessary that the *inside* parts cross the cloth again. If our untangling method runs alongside a traditional collision prevention algorithm such as [3], we disable collision detection for *inside* parts of the cloth, as well as a small region around the intersection lines, allowing for *inside* parts to cross over and correct intersections.

In case an *inside* region does not contain any mesh vertices, we randomly choose an *inside* point on each triangle in the region, and compute the penalty force for these points. The force is then applied to the vertices of the triangle containing the point.

Note that it is also possible to directly resolve an intersection by displacing the *inside* vertices. In order to do so, correspondences are computed for each path pair, and the *inside* vertices are moved somewhat more than half-way to their target points. Since this is done for both paths, this will separate the sheets. This method does not resolve BLI intersections in one step, although applying the correction in an iterative fashion will eventually resolve the intersection.

While direct separation works well for small intersections, there is no guarantee for success, even if applied iteratively. For more complex intersections, this method becomes unstable and may oscillate. In practice, relaxation using penalty forces proved superior in most situations.

5.3 Handling BI/BI Paths

Since BI paths do not define *inside* regions, BI/BI path pairs cannot be handled as described above. Instead, we apply forces that push the intersection towards the nearest boundary.

We define a distance field $h(\mathbf{u})$ in parameter space, storing the distance to the nearest cloth boundary for each point on the cloth. A force in the direction of the closest boundary of the cloth acts in the direction of the negative gradient of h.

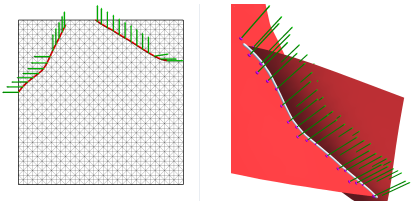
To define this force in world space, we consider the tangent space of the parameterized surface $\mathbf{s}(\mathbf{u})$ with parameterization coordinates $\mathbf{u} = (u, v) \in \mathbb{U}$. The tangent space is spanned by the vectors $\frac{\partial \mathbf{s}}{\partial u}$ and $\frac{\partial \mathbf{s}}{\partial v}$, thus for each point $\mathbf{x} = \mathbf{s}(\mathbf{u})$, the 3×2 matrix

$$T(\mathbf{u}) = \left(\frac{\partial \mathbf{s}}{\partial u} \; \frac{\partial \mathbf{s}}{\partial v} \right) \quad (5)$$

transforms vectors in $\mathbb{U}$ to their corresponding tangent directions in world space. In our case, the surface is defined as a mesh. Since both world space coordinates and (u, v)-coordinates are known for all mesh vertices, $T(\mathbf{u})$ is straightforward to compute.

Using $T(\mathbf{u})$, we define a world space force field $\mathbf{F}_{\text{out}} \in \mathbb{R}^3$ at each point on the cloth:

$$\mathbf{F}_{\text{out}}(\mathbf{u}) = -\frac{T(\mathbf{u})\nabla h(\mathbf{u})}{\|T(\mathbf{u})\nabla h(\mathbf{u})\|} \cdot \max(kh(\mathbf{u}), f_{\max}) \quad (6)$$



(a) (b)

Figure 5: (a) $-\nabla h(\mathbf{u})$ for points on a BI /BI path pair. (b) Forces acting on one of the paths in world space. The intersection is pushed outside the cloth.

Again, the force magnitude grows with penetration depth, up to a maximum value $f_{\max}$.

In order to push an intersection path outwards, we average the forces over the entire path. Consider a BI path $P = \{\mathbf{r}(t)\} \subset \mathbb{U}$ and its corresponding path $P' = \{\mathbf{r}'(t)\} \subset \mathbb{U}$. In order to resolve the intersection, a correction force $\mathbf{F}_P$ is applied to all points $\mathbf{r}(t)$ on the path P. Since the paths are piecewise linear, the average force can be expressed as a sum over the path segments.

$$\mathbf{F}_P = \sum_i \frac{l_i}{l_{P'}} \mathbf{F}_{\text{out}}(\mathbf{r}'(\frac{t_i + t_{i+1}}{2})) \qquad (7)$$

Here, l_i is the length of the path segment i, and $l_{P'}$ is the length of the path P'. We use one force sample per segment, i. e. at least one sample per triangle. At each point, F_{out} is tangent to the surface intersected by the piece of cloth around $\mathbf{r}(t)$. The path P is thus pushed over the surface around P'. Note that similar to the correction proposed in Section 5.2, the forces should be applied symmetrically in order to conserve momentum.

Typically, there are no vertices on the intersection path. We apply the correction force to all vertices of triangles that contain an intersection segment. Figure 5 shows an illustration of the correction forces for BI paths.

Note that this technique is quite similar to the untangling method proposed in [18]. However, we push the intersection towards the closest boundary, while [18] move the intersection segments in the direction that reduces their total length. Both approaches have their pathological cases. We believe that for BI paths, our method leads to more natural untangling behaviour.

6 Results

We have implemented the untangling method as described above and applied it to cloth in various tangled states. Figure 6 shows an example of a LL intersection and its resolution after several steps of relaxation. Figures 7 and 8 on the color page show untangling of a BB /II path pair and a LL path respectively. Note that contrary to [2], we actively correct LL intersections.

Table 1 summarizes computation times on a P4 3GHz. Our path finding code is in no way optimized, and currently needs $O(n^2)$ time instead of the possible $O(n \log n)$, where n is the number of segments. We believe that cloth untangling should be used to complement, not replace collision prevention methods such as [3]. Our method can be used to detect intersections and adjust the collision handling accordingly where collision prevention has failed. Note that this also reduces the load on collision detection, as small errors are now acceptable.

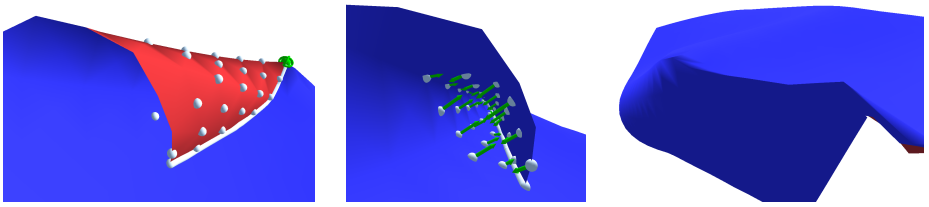
#Tri	#Paths	#Seg	#In	Seg	Path	RBF
2450	4	154	147	32	<1	<1
	23	1976	827	78	281	32
31250	8	284	147	360	15	<1
	30	4540	1515	656	7078	172

Table 1: Computation times of our algorithm. The columns show the number of mesh triangles, the number of intersection paths, the number of intersection segments, and the total number of inside vertices, as well as computation times for finding segments, assembling paths and RBF fits in milliseconds.

7 Conclusion and Future Work

We have presented a thorough analysis of intersection types on 2D manifolds. Compared to [2], using RBF fitting instead of closest point matching for force computation greatly improves robustness of the intersection handling, especially in the presence of foldovers in the intersection region. It also enables intersection handling for path types that were previously impossible to correct. We have also proposed a method for correcting intersections that do not define any inside regions.

Thus, arbitrary intersections of cloth with boundaries can be resolved.



(a) (b) (c)

Figure 6: (a) A BLI intersection. (b) A view inside the BLI cone showing the correction forces. (c) As the cone is flattened, the intersection is resolved (top view).

While our methods handles intersections reliably, extremely complicated intersections can lead to strange unfolding behaviors. On severely tangled cloth, intersection paths can self-intersect, or come extremely close to each other. While our algorithm typically unfolds and resolves the intersections, it might do so in a non-intuitive manner. We also cannot give formal guarantees that the cloth will unfold entirely, as there might be pathological configurations leading to oscillations.

At the moment, most cloth simulations deal with 2D manifolds, thus our method is general enough in these settings. Since most real-life garments are actually non-manifold, it would be interesting to generalize our analysis to nonmanifold surfaces.

Acknowledgements

This project was funded by the Swiss National Commission for Technology and Innovation (CTI) project no. 7560.1 ESPP-ES.

References

[1] D. Baraff and A. Witkin. Large Steps in Cloth Simulation. In *Proceedings of Siggraph '98*, 43–54, 1998.

[2] D. Baraff, A. Witkin, and M. Kaas. Untangling Cloth. In *Proceedings of Siggraph '03*, 862–869, 2003.

[3] R. Bridson, R. Fedkiw, and J. Anderson. Robust Treatment of Collisions, Contact, and Friction for Cloth Animation. In *Proceedings of Siggraph '02*, 594–603, 2002.

[4] R. Bridson, S. Marino, and R. Fedkiw. Simulation of Clothing with Folds and Wrinkles. In *Proceedings of the Symposium on Computer Animation*, 28–36, 2003.

[5] M. D. Buhmann. *Radial Basis Functions*. Cambridge University Press, 2003.

[6] M. Carignan, Y. Yang, N. Magnenat-Thalmann, and D. Thalmann. Dressing Animated Synthetic Actors with Complex Deformable Clothes. In *Proceedings of Siggraph '92*, 99–104, 1992.

[7] K. Choi and H. Ko. Stable but Responsive Cloth. In *Proceedings of Siggraph '02*, 604–611, 2002.

[8] E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. Discrete Shells. In *Proceedings of the Symposium on Computer Animation '03*, 62–67, 2003.

[9] M. Meyer, G. Debunne, M. Desbrun, and A. Barr. Interactive Animation of Cloth-Like Objects in Virtual Reality. *J. of Visualization and Computer Animation*, 12(1):1–12, 2001.

[10] C. A. Micchelli. Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Functions. *Constructive Approximation*, 2(1):11–22, 1986.

[11] N. M. Patrikalakis. Surface-To-Surface Intersections. *Computer Graphics and Applications*, 13(1):89–95, 1993.

[12] X. Provot. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour. In *Proceedings of Graphics Interface '95*, 147–155, 1995.

[13] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically Deformable Models. In *Proceedings of Siggraph '87*, 205–214, 1987.

[14] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross. Optimized Spatial Hashing for Collision Detection of Deformable Objects. In *Proceedings of Vision, Modeling and Visualization '03*, 47–54, 2003.

[15] M. Teschner, B. Heidelberger, M. Müller, and M. Gross. A Versatile and Robust Model for Geometrically Complex Deformable Solids. In *Proceedings of Computer Graphics International '04*, 312–319, 2004.

[16] P. Volino, M. Courchesne, and N. Magnenat-Thalmann. Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects. In *Proceedings of Siggraph '95*, 137–144, 1995.

[17] P. Volino and N. Magnenat-Thalmann. Accurate Collision Response on Polygonal Meshes. In *Proceedings of Computer Animation '00*, 154–163, 2000.

[18] P. Volino and N. Magnenat-Thalmann. Resolving surface collisions through intersection contour minimization. 1154–1159, 2006.

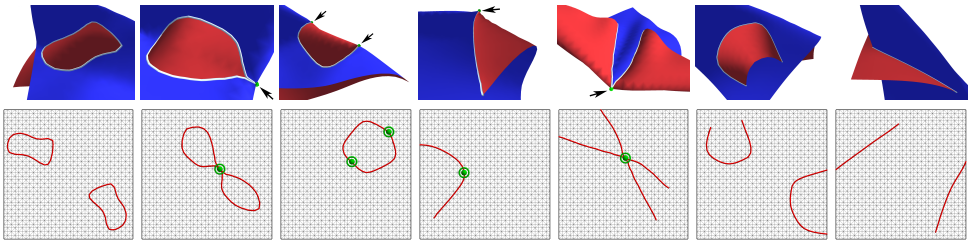


Figure 1: Different paths in world space (top row) and parameter domain $\mathbb{U}$ (bottom row). Loop vertices are marked on the paths. (a) a pair of `CLOSED` paths. (b) `EIGHT` path. (c) loop-loop (`LL`) path. (d) boundary-loop-inside (`BLI`) path. (e) `CROSS` path. (f) boundary-boundary/inside-inside (`BB`/`II`) path pair. (g) boundary-inside (`BI`) path pair.

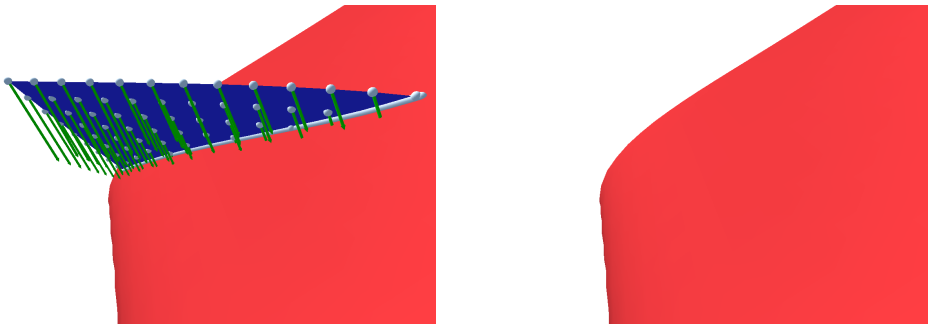


Figure 7: (a) An intersection creating a `BB`/`II` path and the penalty forces applied in this situation. (b) After several timesteps, the intersection is resolved.

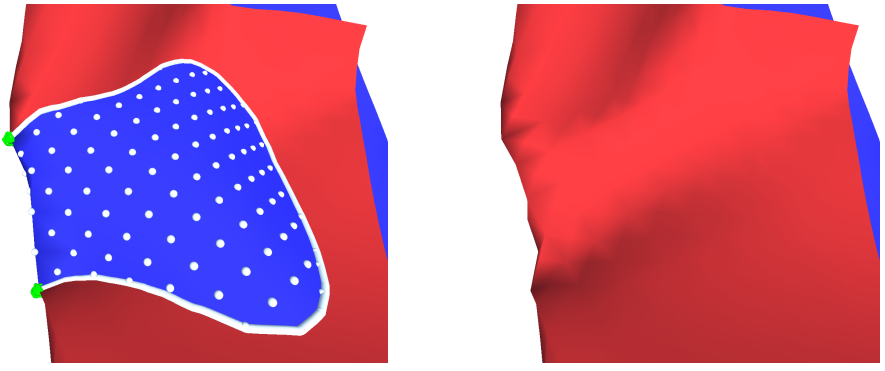


Figure 8: (a) A `LL` intersection path. (b) After several timesteps, the intersection is resolved.