Resolving Surface Collisions through Intersection Contour Minimization

Pascal Volino[*]
MIRALab, University of Geneva

Nadia Magnenat-Thalmann[**]
MIRALab, University of Geneva

Abstract

Robust handling of collisions on non-oriented deformable surfaces requires advanced methods for recovering intersecting surfaces. We present a novel method that resolves intersections between two intersecting surface regions by inducing relative displacements which minimize the length of the intersection contour between them. This method, which does not rely on intersection regions, has a broader application field than existing methods, and its implementation is also much simpler, allowing integration into most existing collision response schemes. We demonstrate the efficiency of this method through examples in the context of cloth simulation.

CR Categories: I.3.7 [Three-Dimensional Graphics and Realism]: Animation, Virtual Reality; I.6.3 [Simulation and Modeling]: Applications.

Keywords: Collisions, Surface Intersections, Cloth Simulation.

1. Introduction

Cloth simulation involves accurate mechanical models and numerical methods that can reproduce the properties of cloth materials and integrate the resulting equations in realistic computation times. However, in many practical applications such as garment simulation, collisions play a major role in the simulation, and have to be handled comprehensively. Putting aside the problematic of detecting collisions between mechanical objects efficiently and their integration in the mechanical model (a good overview of this is described by Teschner et al [2005]), we here focus on robustness issues that are important to address in the context of simulation of complex mechanical surfaces.

In early cloth simulation models, collision processing was restricted to prevent cloth surfaces to penetrate volumes (for instance, a garment surface against a body volume). The surfaces describing the hull of these volumes had a clear "inside-outside" orientation, and collision processing only had to bring the cloth surface on the right (outside) side of the volume hull surface.

In more complex simulation contexts (for instance, simulating a complex set of layered garments on a virtual character, as shown in Fig.1), a comprehensive handling of all possible collisions that may occur in the simulation is needed. This involves detecting collisions between all different surface areas of cloth. Unlike surfaces describing volume hulls, cloth surfaces do not have any orientation (both sides are "outside"), and contacts on any of the sides are valid. Unfortunately, this context prevents a simple resolution of inconsistently intersecting surfaces.

Surface intersections do not usually represent a physically correct state in usual simulation systems. Most of the time, they result from approximate collision detection and response schemes that are not able to enforce consistently the geometrical collision constraints along any situations that may occur during the simulation. Unfortunately, comprehensive methods for ensuring adequate processing of all geometrical collision constraints are complex (detection of numerous mesh collision configurations, interactions between numerous collisions, handling correctly geometrical singularities and numerical errors...). Therefore, they are totally impractical to implement for processing large numbers of collisions with realistic computation times (for example, in the context of simulation of complex garments). Meanwhile, faulty initial geometric configurations and nonphysical factors in the simulation context may also lead to surface intersections (for instance, when the cloth is pinched by two interpenetrating geometric objects, as described by Baraff et al [2003]).

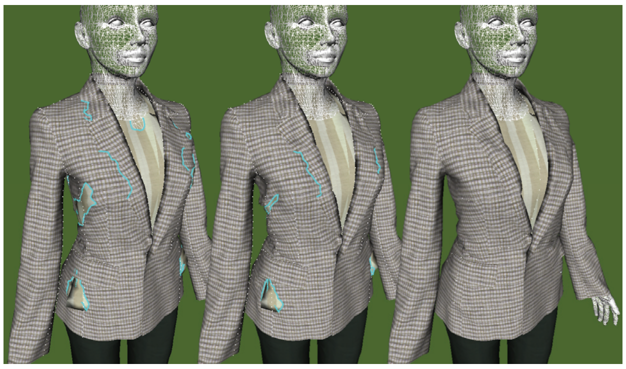


Fig.1. Robust simulation of complex garments requires algorithms that automatically remove of intersections between multiple layers of cloth during the simulation (from left to right). Our approach is based on minimization of the intersection contour length (blue lines).

As we can see, defining a robust simulation system for complex cloth objects cannot only be done by trying to prevent surface intersections from occurring, but also requires being able to "repair" those intersections whenever they occur (Fig.1). Several approaches are available for solving this problem. The most common approach is to identify "collision regions" consisting of adjacent collisions between common surface regions, and to ensure global orientation consistency between the collisions of a region. This approach has been followed by Volino et al. [1995], by defining the global orientation of a collision region by majority

[*] e-mail: pascal@miralab.unige.ch
[**] e-mail: thalmann@miralab.unige.ch

polling the orientations of all collisions of that region, and correcting the orientation of the minority collisions when required. Each collision would also be tracked along time, retaining a constant orientation with detection and correction of orientation changes resulting from surface crossings along time. While fairly efficient in many contexts, such methods are however quite impractical to implement, as they require complex data structures and heavy algorithms for grouping detected collisions into regions, and tracking evolving collisions along time during all the simulation. Furthermore, as discussed by Baraff et al [2003], such history-dependent methods may even prevent recoveries if, for any reason, an erroneous configuration is "remembered" to be correct by the simulation.

Simpler approaches aim at preventing surface intersections to occur or spread, mostly by constraining regions where such situations might happen, removing all instability that might result from interaction with mechanical simulation. For instance, Baraff et al [2003] use "flypapering" for preventing instabilities in intersecting mechanical surfaces, and Bridson et al [2003] use geometrical methods to preserve wrinkle patterns in colliding surfaces. The most comprehensive way of preventing intersections to occur is presented by Bridson et al [2002]. Anyhow, whatever the complexity of these methods, none can prevent intersections to occur if they *have to* because of non-consistent constraints.

A major contribution in the resolution of intersecting surfaces has been brought by Baraff et al [2003] through their Global Intersection Analysis method. This method tracks intersections by reconstructing the intersection contours of the colliding regions. Once a closed contour has been identified, the colliding surface regions within is identified using a "flood-fill algorithm" on the mesh and, once orientation correspondence between the detected regions is established, their collision response method would bring these surfaces back to their non-colliding relative positions as soon as it is possible, in conjunction with their "flypapering" method. However, a major restriction is the necessity to have "well-defined" intersection regions unambiguously identified by intersection contours. Unfortunately, this is not always the case when surface boundaries are involved in the intersection, and we have found that such situations occur quite often, when resulting from approximate collision processing of complex surfaces. Even more, sometimes the "orientation" of intersecting surfaces does not even make sense (for example, in Fig.2 bottom). Hence, methods based on orientation of collision and intersection regions are quite unlikely to work correctly in the contexts described in Section 3.1.1 and 3.1.3.

Our goal is to overcome these limitations through the definition of a new method for resolving surface intersections. Rather than spending time identifying colliding surface regions as done by Volino et al [1995] or Baraff et al [2003], we base our method on a very simple paradigm: Minimizing the length of the intersection contours (Fig.2).

Through these developments, we present a very general scheme for resolving surface intersections that do not suffer from the limitations of collision region and surface orientation reconstruction methods. It can complement most forms of collision response techniques, as long as local surface intersections are detected. We put emphasis on the versatility of the method, and we detail two variations offering various degrees of simplicity and efficiency. None of them require complex algorithms or data structures, and none rely on the history of the simulation. We describe the principles of this method in Part.2, and some examples illustrating its capabilities are shown in Part.3.

2. Description of the Method

In the following, we assume that the surfaces are described as polygonal meshes, and that the polygons are flat (this is always the case for triangle meshes). In this context, regular surface collisions are typically detected as proximities between vertices and polygons, sometimes between edges. These collisions illustrate that surfaces are close enough for considering them as interacting. Meanwhile, surface intersections are typically detected as intersections between edges and polygons. Their presence denotes that surfaces interpenetrate in a usually not physically plausible way, and we aim to present a scheme to resolve this situation. The surface intersection contour is actually described as two lines, drawn identically on the two intersecting surfaces. In the case of polygonal meshes, this contour is indeed a polygonal line, and the edge-polygon intersections define the vertices supporting it (Fig.2).

The idea of resolution scheme is fairly simple: We define a collision response scheme that induces a relative displacement between intersecting edges and polygons so as to reduce the length of the intersection contour, ultimately leading to the disappearance of the surface intersection.

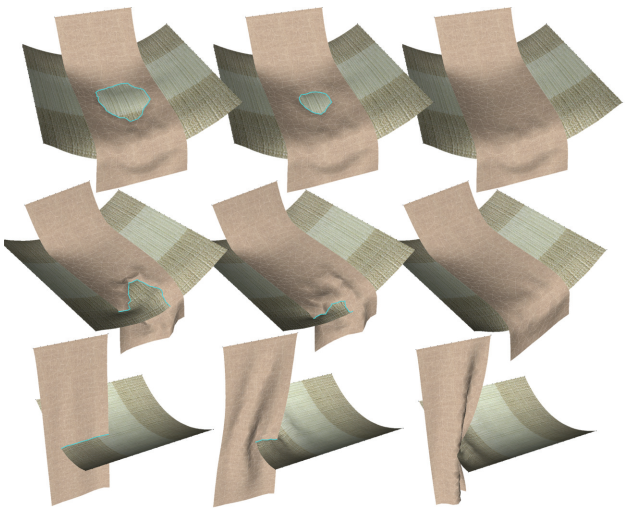


Fig.2. Minimizing the intersection contour length not only resolves the case of closed contours (top), but also open contours (middle) and cases where the orientation of "collision regions" is meaningless (bottom).

2.1. Minimizing Edge-Polygon Intersections

In a mesh, an edge **E** is typically adjacent to several polygons $\mathbf{B_i}$ (usually two, only one if the edge is on a surface boundary). When this edge intersects a polygon **A**, the intersections between **A** and $\mathbf{B_i}$ define segments of the intersection polygonal line, drawn on the surfaces of **A** and $\mathbf{B_i}$ in an identical way. We initially focus on how a small displacement **D** of the edge **E** relatively to the polygon **A** changes the length of the intersection segment between *one* polygon $\mathbf{B_i}$ and the polygon **A**.

Naming **N** and $\mathbf{M_i}$ the normals of the polygons **A** and $\mathbf{B_i}$ respectively, we first compute a vector $\mathbf{R_i}$ representing the direction of the intersection segment. This vector is obtained through cross-product of **N** and $\mathbf{M_i}$. We normalize it to unit length and choose its orientation so as to indicate from the edge **E** the *inside* of the polygon $\mathbf{B_i}$ (Fig.3). Using the geometric construction shown in Fig.3, with **E** representing any direction vector of the edge, and noting that **E** is collinear to the vector $\mathbf{P_bQ}$ and **N** is orthogonal to the vector $\mathbf{P_aQ} = \mathbf{D}+\mathbf{P_bQ}$, we have:

$$(N \cdot E)\,\overline{P_bQ} = \left(N \cdot \overline{P_bQ}\right) E = -(N \cdot D)\,E \qquad (1)$$

From this, the reduction of intersection segment length $\mathbf{l_i}$+$\mathbf{k_i}$ drawn on the polygon **A** (as the intersection point slides from $\mathbf{P_a}$ to **Q** over it) is computed as following:

$$l_i + k_i = \overline{P_aQ} \cdot R_i = D \cdot R_i + \overline{P_bQ} \cdot R_i = D \cdot R_i - \frac{N \cdot D}{N \cdot E} E \cdot R_i \qquad (2)$$

Meanwhile, the reduction of intersection segment length $\mathbf{k_i}$ drawn on the polygon $\mathbf{B_i}$ (as the intersection point slides from $\mathbf{P_b}$ to **Q** over it) is computed as following:

$$k_i = \overline{P_bQ} \cdot R_i = -\frac{N \cdot D}{N \cdot E} E \cdot R_i \qquad (3)$$

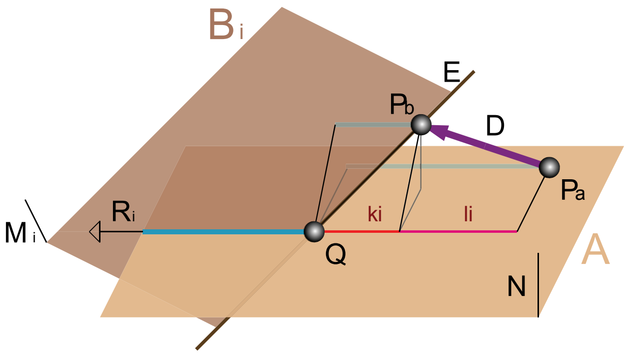


Fig.3. When the edge **E** moves by a small displacement **D** relatively to the polygon **A**, the intersection point between **E** and **A** moves from $\mathbf{P_a}$ to **Q** over the polygon **A**. Thus, the length of the intersection segment of $\mathbf{B_i}$ drawn on **A** is shortened by $\mathbf{l_i}$+$\mathbf{k_i}$, Meanwhile, the intersection point slides from $\mathbf{P_b}$ to **Q** over the polygon $\mathbf{B_i}$. Thus, the length of the intersection segment of **A** drawn on $\mathbf{B_i}$ is shortened by $\mathbf{k_i}$.

We need to evaluate the change of length of the contours drawn on *both* intersecting surfaces. For this, we can now add the contributions of all polygons $\mathbf{B_i}$ adjacent to the edge **E**. The total length reduction of all intersection segments drawn on polygons **A** and all $\mathbf{B_i}$ can be expressed from the relative displacement **D** through a linear gradient vector **G**, as follows:

$$\sum_i (l_i + 2\,k_i) = G \cdot D \qquad \textit{where} \qquad G = \sum_i \left(R_i - 2\,\frac{E \cdot R_i}{E \cdot N} N \right) \qquad (4)$$

The direction of the gradient vector **G** indeed represents the direction along which the edge **E** should be moved relatively to the polygon **A** so as to get the best reduction of the intersection contour length (Fig.4). Hence, with a relative displacement **D** between both colliding elements, the intersection contour length reduction is **G·D**, assuming a first-order approximation that preserves the directions of the edge **E** and the intersection $\mathbf{R_i}$ (pure translation between colliding elements), and that the intersection point does not "jump out" of the edge **E** or the polygon **A**. In practice, this justified as long as the displacement **D** does not become much larger compared to the size of a mesh element (such requirement is quite typical to most collision response schemes), since surfaces are most of the time regular enough to have similar intersection angles when the contour moves between adjacent polygons. In these conditions, the length of the gradient still represents roughly how efficiently a relative displacement along its direction reduces the length of the intersection contour. It can be noted that **G** is always null if the intersected edge separates two polygons of a flat surface.

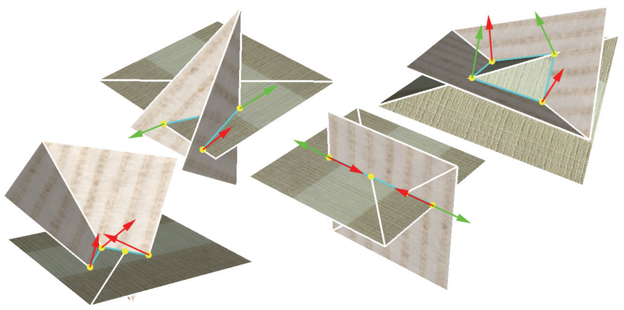


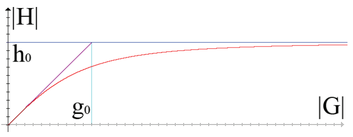
Fig.4. Some configurations representing edge-polygon intersections (yellow dots), and the orientation of the gradient **G** (red arrows) or its opposite (green arrows). Reducing the length of the intersection contour is done by moving intersected edges along the red arrows and intersected polygons along the opposite green arrows.

It is important to note that there are no assumptions on the number of polygons $\mathbf{B_i}$ adjacent to the edge **E**. Hence, the presented method is perfectly applicable for non-manifold meshes. This can be important in the field of garment simulation, when complex seams are involved in multiple collisions (Fig.1).

2.2. Integration with Collision Response

Collision response aims at integrating the effects of collisions in the simulation. In the context of mechanical simulation, this is possible through ad-hoc mechanical "penalty forces" modeling the mechanical contact. It can also be done directly through geometrical corrections that enforce geometrical and kinematical constraints through displacement values, speed or acceleration momentums depending on the actual collision response method, which are then distributed on colliding elements according to mechanical momentum conservation laws. In all cases, the direction of the gradient **G** and its amplitude are good indications of the direction to follow for correcting the intersection.

Integration of our intersection resolution in such a collision response scheme is done by defining an "interaction vector" **H** from the gradient **G** according to the features and possibilities of the collision response scheme. Obviously, we should construct the vector **H** on the same direction as the vector **G** for optimizing the intersection contour length reduction. Then, their norm should be related as well, but adequate solutions highly depend on the nature and properties of the collision response scheme. As a fairly general solution, we propose to use the following scaled arctangent function:



$$|H| = h_0 \frac{|G|}{\sqrt{|G|^2 + g_0^2}} \qquad (5)$$

The value $\mathbf{h_0}$ represents the maximum norm of the correction acceptable for the collision response scheme, while the value $\mathbf{g_0}$ defines a progressive slope that stabilizes the response if the intersection contour length minimization is ambiguous.

2.3. The Local and the Global Scheme

Implemented as is, the intersection correction scheme handles all edge-polygon intersections independently, locally reducing the size of the intersection contour around them. This local scheme is indeed very simple to implement, as only the geometry of the colliding mesh elements is needed, without any need to explore the surface mesh further away. While this scheme is able to solve most surface intersection configurations, it is however sometimes not very efficient. Hence, while small or closed intersection contours on quasi-parallel surfaces (Fig.2 top) are resolved efficiently, longer and complex contours (Fig.2 middle) require significantly more iterations. Indeed, the iterative scheme of the contour length reduction process is fairly similar to a "snake" that reduces its length by locally straightening itself. This is not efficient if the contour is already quite straight, and actions on the extremities of open contours are frequently the largest contributors for the length reduction. In the particular context of "straight" open contours (Fig.2 bottom), only the two edge-polygon intersections at the extremities of the contour actively participate to the length reduction.

The idea of the global scheme is to have all the corrections of a given intersection contour to "work together" towards a global reduction of the contour length. For this, we add all gradients corresponding to all edge-polygon intersections of a given intersection contour to a single total gradient (their signs might be flipped depending on which surface carries the intersecting edge and which carries the intersecting polygon). This total gradient represents how the total length of the intersection contour varies respective to the relative displacement between the intersecting surfaces. The collision correction resulting from this total gradient is then applied uniformly over all edge-polygon intersections involved in the contour (Fig.5).

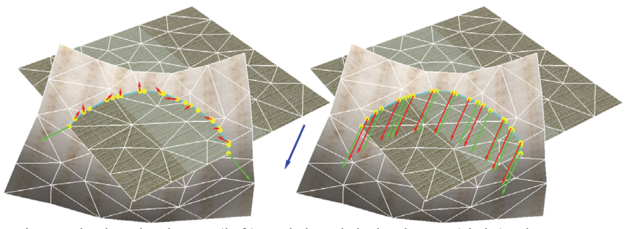


Fig.5. The local scheme (left) and the global scheme (right), shown on a given surface intersection. With the local scheme, correction is applied on each intersection according to its own gradient. This is not very efficient if the intersection contour is almost straight at each side of the edge-polygon intersections. With the global scheme, correction is applied identically on all intersections according to the sum of all gradients in order to globally reduce the length of the whole contour.

It can be noted that it is not essential to implement a very robust and complete intersection contour reconstruction algorithm for having the global scheme to work. First, it is only necessary to group the detected edge-polygon intersections respectively to the intersection contour they belong to, not to order them so as to have the contour polygonal line explicit. Then, any edge-polygon intersection missing from the collision detection will not break the algorithm (a contour could be split into several sections handled independently). Thus, we can highly simplify the collision detection algorithm by just ignoring geometric singularities and special intersection cases (such as edge-edge or polygon-vertex intersection coincidences). Therefore, our algorithm can deal quite robustly with imperfect and approximate collision detection methods, as demonstrated in Section 3.1.2.

3. Results

We acknowledge that the efficiency of the presented method is likely to be very dependent on the actual nature and implementation of the simulation and collision response system in which it is actually integrated. We will however illustrate this efficiency using a mechanical cloth simulation system which is fairly typical to the implementations actually used.

This method has been integrated in a cloth simulation engine based on particle systems tuned for accurate representation of the mechanical properties of cloth, based on the ideas described by Etzmuss et al [2003] and Volino et al [2005]. Numerical integration is performed using Implicit Euler, as described by Baraff and Witkin [1998] and Hauth et al. [2002]. Broad-phase collision detection is performed using Axis-Aligned Bounding Volume hierarchies using Discrete Orientation Polytopes from Klosowsky et al [1997], and self-collision detection optimizations from Volino et al [1994]. Collisions are then refined by detecting proximities between mesh elements below a given threshold distance for regular collision response. Meanwhile, surface intersections are detected through edge-polygon intersections. Collision response is performed by enforcing the collision constraints through geometric corrections of position, speed and acceleration distributed on mesh elements so as to preserve mechanical momentum, in a similar way to Volino et al [2000], without any subframing. Hence, one *iteration* of the computation refers to one mechanical iteration with collision response, which also considers the gradients of the intersection resolution scheme.

We illustrate the efficiency of our method through a set of challenging examples. None of these examples contain surfaces with predefined surface orientation information.

3.1. Some Test Examples

3.1.1. Untangling Cloth Surfaces

In this test, we let fall 25 squares of cloth on a ground (about 300 polygons each). These squares are initially highly intersecting, producing more than hundred mangled intersection contours (roughly 2000 edge-vertex intersections) (Fig.6). Without intersection resolution, the whole set falls like a clump on the ground without any improvement in the collisions. With resolution, intersecting squares are adequately pushed apart, quickly leading to a totally intersection-free configuration. The simulation was computed between 0.4 and 0.2 second per iteration on a 3GHz Intel Pentium4 PC (it depends on the amount of "regular" collisions), and the extra computational cost required by our method was not measurable.

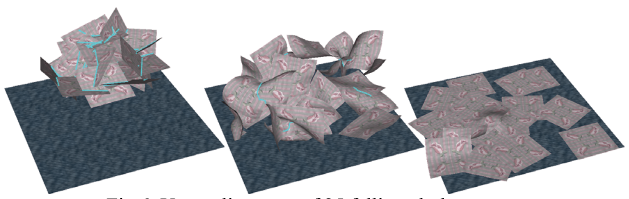


Fig.6. Untangling a set of 25 falling cloth squares.

The simulation has been carried out with the global scheme. Despite complexity, a large amount of intersections are resolved in the first 10 iterations. The local scheme works poorly in this context, because of the extent of the intersection contours.

3.1.2. Robustness Test

A major issue in the lack of robustness results from incomplete collision detection, as designing comprehensive algorithms for detecting all possible geometric configurations is quite impractical. We have simulated a faulty collision detection system by randomly ignoring 50% of all collisions (regular collisions as well as intersections) detected by our system. In the cloth simulation context shown in Fig.7, this produces small intersections randomly appearing between colliding surfaces.

Using intersection resolution, intersecting regions are resolved a few frames after they appear, and the simulation is maintained along time. Without, they quickly expand, breaking the simulation.

While the local scheme is fairly efficient in this context as long as intersecting regions are not allowed to expand too much, the global scheme gives better robustness in more difficult cases, despite the fact that ignoring some detected intersections prevents intersection contours from being completely reconstructed. Most intersections are resolved in only 1 or 2 frames. The simulation, which involves about 2400 polygons, is simulated at about 12 iterations per second on a 3GHz Intel Pentium4 PC, with no measurable slowdown resulting from intersection correction.

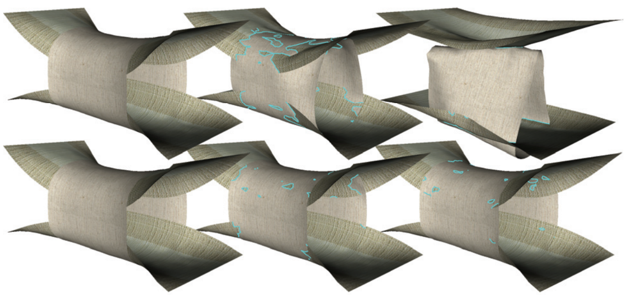


Fig.7. Cloth simulation (frames 0, 10, 20 from left to right) while randomly ignoring 50% of the detected collisions: Without (top) and with (bottom) Intersection resolution.

3.1.3. The Ribbon Simulation

This torture-test scenario for collision processing consists in a very long ribbon (120 meters long × 5 centimeters wide) falling on a rotating and slowly tilting dish (Fig.8). Using our implementation, the computation of the ribbon, which has about 80000 polygons, took roughly between 3 and 30 seconds per iteration on a 3GHz Intel Pentium4 PC, depending on the highly variable amount of collisions.

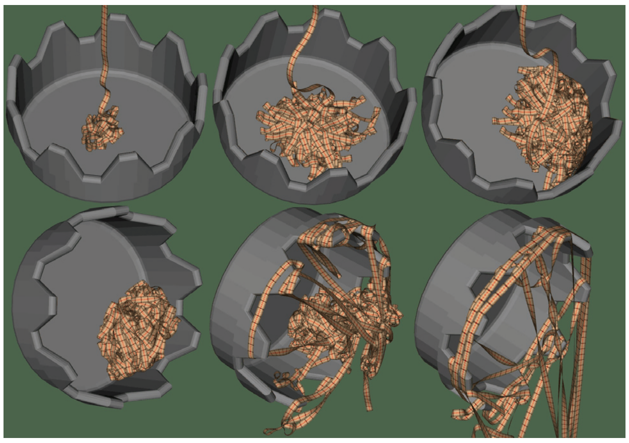


Fig.8. The ribbon falling in the rotating dish.

Despite its performance, collision response is unable to ensure no intersections among the several tens of thousands of highly interacting collisions detected in each frame. This is particularly true when the ribbon "pasta" rolls in the tilted dish (Fig.8 bottom left), because of all the sliding produced by the shear deformation of the rolling, and a few intersections always come to happen in each frame. Without any intersection resolution method, these quickly accumulate and propagate in the system, producing inconsistent ribbon "crumple clumps" that finally break the simulation.

Using our intersection resolution method, these intersections are removed only a few frames after their occurrence, ensuring correct untangling of the ribbon at the end of the simulation. The local scheme was sufficient to address this context, as intersections usually don't extend too much as they appear.

3.1.4. Garment Simulation

This intersection resolution system is actually being used in garment simulation. This is a particularly important feature when the aim is to design complex dressings made of several layers of cloth (Fig.1), and when these are animated using geometrically-deformed bodies that are not always mechanically accurate. In this context, the interest of the intersection resolution method is two-fold: First, it ensures that the simulation always reverts back to correct even when any simulation artifact or inaccuracy produces intersections. Then, it greatly enhances the ease of garment design since an exact initial placement of garments is not necessary (Fig.1). The global scheme offers best results in this case, as the intersecting regions might be quite extended.

3.2. Performance and Limitations

Computationally, the extra processing required by our method is very low, and in practice not measurable compared to the processing of the "standard" collision detection and response algorithms in which it has been implemented. This is mainly because the number of edge-polygon intersections remains very low compared to the number of regular collisions between mesh elements. Possibly, most additional computation would result from the explicit detection of polygon-edge intersections, a task which should be performed anyhow for detecting that surfaces do intersect.

Resolution of the surface intersections is an iterative process, and the iteration "width" is inherently dependent on the amount of "correction" that the underlying collision response scheme can afford. Our tests have also shown that it is much more efficient to resolve intersections progressively along the simulation (with a very limited number of resolution iterations between each mechanical iteration) than attempting to completely correct all intersections before next simulation step. This allows regular simulation to relax and fair the correction artifacts progressively along the intersection correction process, and eventually jump out of any possible local minimum that result from particular geometric configurations. Most cases (such as those shown in Figs.1, 2, 7) are typically resolved in less than ten iterations with our implementation.

Unfortunately, there are still a few intersection configurations where this minimization cannot remove the intersection. Some of them are shown in Fig.9. Our algorithm then converges to a local minimum of the intersection contour length (Stability of this convergence can be obtained with a sufficiently large value of $\mathbf{g_0}$ in formula (5)). These configurations represent the cases where the paradigm of reducing the length of the intersection contour is not sufficient alone for "improving" the current surface intersections toward full resolution. Not all of these configurations would be solved systematically by any other method anyway, as sometimes there is no clear and formal way to devise from them what would actually be the non-intersecting configuration (one could consider the largest side respective to the intersection contour, the shortest path to pull surfaces apart, not ignoring geometrical constraints such as attach points, the simulation context and high-level metadata or ontology...). For instance, a possible addition to the presented method would be to identify intersection regions whenever possible, and combine the intersection contour length reduction criterion with an intersection area reduction criterion for evaluating the relative displacement between intersecting surfaces. Possibly also, these areas, when well-defined, could be handled directly with a method based on Baraff's ideas [2003].

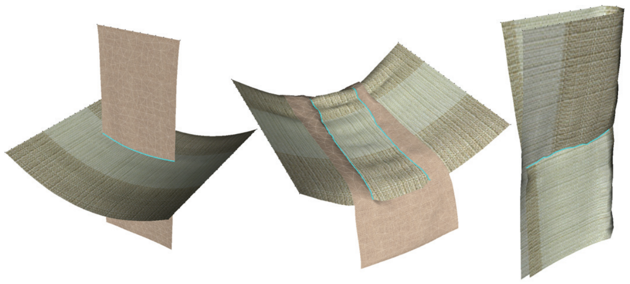


Fig.9. Some particular surface intersection configurations that intersection contour minimization cannot resolve.

In practice, our method also often fails to address particular local crumpling patterns that involve several mangled self-intersecting shapes. In that case, intersection contours could be moved away from their "dead-end" vertices within the surface through local surface-flattening methods. Combination with Bridson's ideas [2002] and other history-based methods through self-collision processing of Govindaraju et al [2005] would also perform well here, since these crumple configurations often result from local surface crossings around excessively bent mesh edges.

Addressing extensively all these cases would anyhow require a combination of several context-specific methods. Anyway, we still believe that intersection contour minimization is a fairly general method that deals with most cases encountered in the context of cloth simulation, while being very simple to implement and combine, if needed, to other specific methods aimed at addressing such particular configurations related to the actual context.

4. Conclusion

We have proposed a new method aimed at resolving surface intersections during animation processes. Among its major interests, it is very general and does not suffer from the limitations of existing methods based on collision regions and orientation. It is particularly adapted to the context of cloth simulation, and may resolve most of the challenging cases encountered in this context.

Another major advantage of this method is the simplicity of its implementation. Hence, the local scheme only relies on a very simple geometrical computation done independently on each detected edge-polygon intersection on the meshes describing the surface, while the global scheme only additionally requires grouping these intersections by intersection contours using algorithms that do not need to be fully comprehensive. The global scheme can efficiently resolve very large surface intersections, and should therefore be favored as a basis for general-purpose cloth untangling. The local scheme is however most of the time sufficient to address small intersections that do not spread far in the mesh, such as those resulting from approximate or faulty collision response, and it is therefore a very simple addition for adding extra robustness in an existing collision processing scheme. We have demonstrated the efficiency of this method through several challenging examples.

Thanks to its generality and simplicity, this method can easily be combined with most usual approaches for collision detection and response on mechanical surfaces, and may also be complemented by other intersection resolution schemes for broader context of use or better performance in specific situations. We are now looking forward to extend this method towards even more challenging goals, such as untangling severely crumpled cloth...

Acknowledgements

We are grateful to all MIRALab team who participated to this research through their contributions, advices, testing and creative artwork. Special thanks for Christiane Luible for her expertise in garment design and animation. This research is funded by the European FP6 project LEAPFROG IP (NMP2-CT-2005-515810).

References

BARAFF D., WITKIN A., 1998, Large Steps in Cloth Simulation, *Proceedings of ACM SIGGRAPH 98*, ACM Press, 32, p.106-117.

BARAFF D., WITKIN A., KASS M., 2003, Untangling Cloth, *ACM Transactions on Graphics* (ACM SIGGRAPH 2003), 22, pp 862-870.

BRIDSON R., FEDKIW R., ANDERSON J., 2002. Robust treatment of collisions, contact, and friction for cloth animation, *ACM Transactions on Graphics* (ACM. SIGGRAPH 2002), pp 594–603.

BRIDSON R., MARINO S., FEDKIW R., 2003, Simulation of Clothing with Folds and Wrinkles, *Eurographics-SIGGRAPH Symposium on Computer Animation*, pp 28-36.

ETZMUSS O., GROSS J., STRASSER W., 2003, Deriving a Particle System from Continuum Mechanics for the Animation of Deformable Objects, *IEEE Transaction on Visualization and Computer Graphics*, 9(4), pp 538-550.

GOVINDARAJU N.K., KNOTT D., JAIN N., KABUL I., TAMSTORF R., GAYLE R., LIN M., MANOCHA D, 2005, Interactive Collision Detection between Deformable Models using Chromatic Decomposition, *ACM Transactions on Graphics* (ACM SIGGRAPH 2005), 24(3), pp 991-999.

HAUTH M., ETZMUSS O., STRASSER W., 2002, Analysis of Numerical Methods for the Simulation of Deformable Models, *The Visual Computer*, 19(7-8), pp 581-600.

KLOSOWSKI J.T., HELD M., MITCHELL J.S.B., 1997, Efficient Collision Detection Using Bounding Volume Hierarchies of k-dops, *IEEE transactions on Visualization and Computer Graphics*, 4(1), pp 21-36.

TESCHNER M., KIMMERLE S., HEIDELBERGER B., ZACHMANN G., RAGHUPATHI L., FUHRMANN A., CANI M.P., FAURE F., MAGNENAT-THALMANN N., STRASSER W., VOLINO P., 2005, Collision Detection for Deformable Objects. Computer Graphics Forum, 24(1), pp 61-81.

VOLINO P., MAGNENAT THALMANN N., 1994, Efficient Self-Collision Detection on Smoothly Discretised Surface Animations using Geometrical Shape Regularity, *Computer Graphics Forum* (Proceedings of EuroGraphics 1994), 13(3), pp 155-166.

VOLINO P., COURCHESNE M., MAGNENAT-THALMANN N., 1995, Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects, *Proceedings of ACM SIGGRAPH 95*, ACM Press, pp 137-144.

VOLINO P., MAGNENAT-THALMANN N., 2000, Accurate Collision response on polygonal Meshes, *Computer Animation 2000*, IEEE Computer Society, pp 154-163.

VOLINO P., MAGNENAT-THALMANN N., 2005, Accurate Garment Prototyping and Simulation, *Computer-Aided Design and Applications*, CAD Solutions, 2(5), pp 645-654.