Collision and self-collision handling in cloth model dedicated to design garments

Xavier PROVOT
Institut National de Recherche en Informatique et Automatique (INRIA)
B.P. 105, 78153 Le Chesnay Cedex, France
Xavier.Provot@inria.fr

Abstract

This article presents a method for collision handling applied to the semi-rigid mass-spring cloth model formerly described in [Pro95].

This method deals with the four main difficulties encountered in collision handling. The first is collision detection. The second is optimization of collision detection, which is otherwise excessively time consuming. The third is collision response. The fourth is conservation of collision consistency. The latter is discussed in detail, and related to cases of interfering multiple collisions. An original method for computation of collision response in this case of multiple collisions is presented, providing a robust conservation of collision consistency.

Results obtained with this approach, in the case of building real garments on a mannequin, are presented and validate our cloth model and collision handling method.

Introduction

Collision handling was first considered in the case of colliding *rigid* objects [MW88, Bar90]. However, cloth animation required the study of the more general case of collisions between deformable surfaces. Cloth models, like our semi-rigid mass-spring model [Pro95], describe the inherent mechanical behavior of the matter of woven fabrics, when submitted to forces of various nature. However, they do not include forces designed to avoid collisions as in [TPBF87] or in [LMTT91]. More generally, it does not tackle the problem of *contact*.

The phenomenon of contact is of a completely different nature from internal mechanical behavior, and it is natural to handle it with a different method. In [CYMTT92], a new method inspired from "inverse dynamics" methods is proposed for collision response computation. This method applies the macroscopic Coulombian laws of friction to the case of a cloth model. This

is much more adapted than using an artificial repulsing force. It is also later used in [LKC96].

But in order to handle collision response, the colliding elements of the meshes of the colliding objects must be detected. The main problem of this collision detection is that it requires a very important computation time.

There are different ways to carry out this collision detection, so as to reduce computation time. Some methods can just locate *at each iteration* the regions where meshes interpenetrate, and then modify these regions so that interpenetration is avoided, as in [VMT94]. The advantage is that really fast algorithms can be implemented in this case. The drawback is that if objects have a very high velocity and if the time-step is too large, some objects may cross each other completely while no collision will have been detected.

Another way is to detect whether collisions occur or not *during each time-step interval*, as in [LMTT91, LKC96]. This is more time-consuming but more accurate. Optimizations are however possible, and some are presented in this article.

Finally, another problem of collision handling is that methods always consider *individual* collisions between two elements of the meshes implied. They do not handle all simultaneous collisions as a whole. As explained in this article, this leads to collision inconsistency, *i.e.* collision detection and response computation as described above does not succeed in avoiding all interpenetration. This is not only due to numerical inaccuracies (as explained in [VMT95]), but also to the fact that multiple collisions may interfere with each other, and the individual treatment of these collisions is not sufficient to solve them, as will be detailed later.

However, Volino *et al* [VMT95, VCMT95] propose an interesting and efficient method for solving this collision consistency problem. This present article presents an alternative method, which circumscribes zones where these multiple interfering collisions occur, and handle them in a specific way to solve completely the collision response problem.

This paper will be structured as follows: we will first present our collision detection method, then the optimizations of this detection, we will explain how we tackle the simple collision response problem, and finally the multiple collision response problem, in order to keep collision consistency.

1 Collision detection

The general case of collision handling is the one involving a cloth object and another moving object of the scene. A particular case is the case of self-collisions, *i.e.* collisions of the deformable cloth object with itself. Regarding both detection and response, both cases are basically handled in the same way, the only difference lies in the optimization of self-collision detection described in 2.2. Therefore, in this section we will not make any difference between self-collision and collision between two different objects. Also, we will only deal in this paper with objects represented by a set of triangles.

Let t_0 be an instant when there is no interpenetration between the cloth and the object. Consider a time interval $[t_0, t_0 + \Delta t]$. Knowing the positions and velocities of each node of our model at time t_0, it is possible to compute its position at time $t_0 + \Delta t$. Collision detection then consists in finding out if one or more collisions occurred during this interval.

These collisions can be of two types:

- either a node of one of the mesh went through a triangle of the other mesh ("point-triangle" collision);
- or the edge of a triangle of one of the mesh went through another edge of the other mesh ("edge-edge" collision).

Note that the numerical integration used in our model is the explicit Euler method (see [Pro95] for more details). The approximation of this integration is that, during the interval $[t_0, t_0 + \Delta t]$, each node moves at a *constant* velocity. This feature is very important for our collision detection method.

1.1 "Point-triangle" collision

Let $P(t)$ be the moving point, and $A(t)$, $B(t)$, $C(t)$ the vertices of the moving triangle. Let also $\overrightarrow{V}$, $\overrightarrow{V}_A$, $\overrightarrow{V}_B$, $\overrightarrow{V}_C$ be their respective constant velocities during $[t_0, t_0+\Delta t]$. We have of course: $A(t) = A(t_0)+t\overrightarrow{V}_A$, $B(t) = B(t_0)+t\overrightarrow{V}_B$, $C(t) = C(t_0) + t\overrightarrow{V}_C$.

If there is collision, then the point $P(t)$ will belong to the triangle $ABC(t)$. This can be written using the following relation:

$$\begin{array}{c} \exists\ t \in [t_0, t_0 + \Delta t] \text{ such that} \\ \exists u, v \in [0,1],\ u + v \leq 1,\ \overrightarrow{AP}(t) = u\overrightarrow{AB}(t) + v\overrightarrow{AC}(t) \end{array} \quad (1)$$

Unfortunately, this vectorial equation yields a *non linear* system of equations. In order to solve this system, another condition expressing that point P belongs to ABC can be used. Indeed, since the vectorial product $\overrightarrow{N}(t) = \overrightarrow{AB}(t) \wedge \overrightarrow{AC}(t)$ is perpendicular to the plane of triangle ABC, the following relation will be satisfied at the time of collision:

$$\overrightarrow{AP}(t) \cdot \overrightarrow{N}(t) = 0$$

This new relation is necessary, though *not sufficient*: it only means that A, B, C and P are coplanar. It is nevertheless useful since it allows the determination of collision time t in a straightforward way. $\overrightarrow{N}(t)$ is a t^2 term, $\overrightarrow{AP}(t)$ is a t term, and their dot product yields therefore a third degree equation that can be solved easily. Three values of t can yet be obtained, among which only those belonging to the interval $[t_0, t_0 + \Delta t]$ can correspond to a collision.

In order to check whether these values of t really correspond to a collision, and not only to coplanarities, they are injected back in equation 1 — which then becomes a *linear* system —.

If several values of (t, u, v) are solutions to the system, the only collision that we must consider is the one that occurred the soonest, *i.e.* the one corresponding to the smallest value of t.

1.2 "Edge-edge" collision

What is concerned here is the detection of a collision, during interval $[t_0, t_0+\Delta t]$, between an edge of the cloth and an edge of the moving object.

Let $AB(t)$ be the first edge and $CD(t)$ be the other one. This time, there will be collision if and only if:

$$\begin{array}{c} \exists\ t \in [t_0, t_0 + \Delta t] \text{ so that} \\ \exists u, v \in [0,1],\ u\overrightarrow{AB}(t) = v\overrightarrow{CD}(t) \end{array} \tag{2}$$

Like before, this leads us to a non linear system. Another relation can nevertheless be used once more in order to find out the value of t without solving the general system above. At the time of collision indeed, the four point A, B, C, D will also lie in a same plane, which can be written:

$$(\overrightarrow{AB}(t) \wedge \overrightarrow{CD}(t)) \cdot \overrightarrow{AC}(t) = 0 \tag{3}$$

This relation yields once again a third degree equation, and allows to compute u and v after having injected t in equation 2. It can thus be detected whether a collision occurred or not.

2 Collision detection optimization

2.1 Bounding boxes hierarchy

Collision handling, and especially collision detection, is the most time-consuming part in cloth animation. Indeed, the collision detection between a cloth model with N mass points and an object of the scene with M nodes has a $\mathcal{O}(MN)$ complexity. The self-collision detection has a $\mathcal{O}(N^2)$ complexity. As soon as we must deal with significantly discretized meshes, this complexity becomes very limitative, and there is a need to reduce it.

We implemented a first simple optimization which consists in dividing the piece of fabric recursively in zones imbricating with each other. The criterion for this recursive partitioning of the triangles of the cloth object is their position in the 2D texture space. At each iteration, a bounding box of these zones can be computed. Then, the collision detection algorithm can be significantly improved by parsing the bounding box tree while eliminating rapidly collisions tests between elements that belong to two zones whose bounding boxes do not intersect. In order to be accurate, the bounding box of each zone does not only bound the position of the zone at iteration $t_0 + \Delta t$, but both its positions at t_0 and $t_0 + \Delta t$.

2.2 Surface curvature and self-collision detection

In the case of self-collision detection, another optimization, inspired from [VMT94], has been implemented.

This optimization is based on the following property: when a given zone (provided it is connex[1]) has a sufficiently "low curvature", it cannot self-intersect, and all the zones it includes do not intersect with each other.

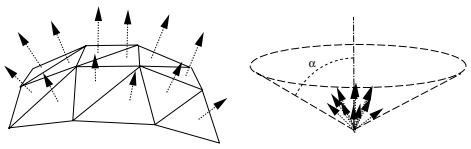


Figure 1: Cone including normals to triangles of a zone of the cloth surface.

The "curvature" of a zone will be in our case evaluated by the set of normals of the triangle belonging to the zone (figure 1). We compute a cone which includes these normals, and the angle α at its vertex is sufficient to build a test that discriminates zones that cannot self-intersect and zones that may: if $\alpha < \pi$, the zone *cannot* self-intersect.

Cones are computed using the hierarchical tree described in previous section (the tree is therefore *not reconstructed* at each time step).

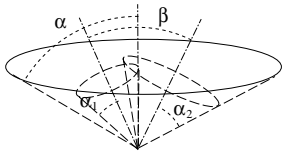


Figure 2: Cone (angle α) enclosing its two "descendant" cones in the hierarchical tree (angles α_1 and α_2).

At the bottom of the tree, each leaf node has a single normal, and therefore $\alpha = 0$, the axis vector of the cone is the normal of the triangle. Then for each tree node for which the cones of its two descendants are known, the cone is computed using the two angles of the descendant cones, α_1 and α_2, and the angle β between the two axes of the descendant cones. The axis vector is computed as the mean vector of the two axis vectors of the descendant cones. The new angle α is then computed as: $\alpha = \beta/2 + \max(\alpha_1, \alpha_2)$ (figure 2).

This is of course only valid if the two descendant zones of each node are *adjacent*. The hierarchical tree described in section 4.4 verifies this property in most cases. Cases of non-adjacency would only occur if the 2D contour of the cloth object were severely non-convex. It never happened in our simulations, even when modeling clothes using real clothes patterns (see section 4.4).

With this technique, it is possible to avoid unnecessary self-collision tests in whole branches of the tree provided they correspond to a zone with

[1] As mentioned in [VMT94], this condition is theoretically not sufficient. It always has been sufficient in practice in the cases we needed to model.

a sufficiently low curvature. At the beginning of an animation for instance, if the piece of cloth is almost flat, no self-collision tests are computed at all.

3 Collision response

3.1 Contact and friction

When two objects collide, there is a time at which they are in *contact*[2]. General macroscopic laws of *friction* describe the forces that are applied to each of the objects when they are in contact. These Coulombian laws can be written as follows.

Consider a mass point P in contact with a motionless rigid surface, at a point H of this surface. Let $\vec{N}$ be a unit normal of the surface at point H. Let $\vec{F}$ be the force applied to P in order to keep the contact. Let $\vec{F}_N = (\vec{F} \cdot \vec{N})\vec{N}$ be the component of $\vec{F}$ perpendicular to the surface and $\vec{F}_T = \vec{F} - \vec{F}_N$ its tangential component.

The laws of friction are:

- if $\|\vec{F}_T\| \geq k_f \|\vec{F}_N\|$, there is sliding contact, with friction, *i.e.* the point moves parallel to the surface, under the action of a force $\vec{F}_s = \vec{F}_T - k_f \|\vec{F}_N\| \vec{u}_T$, where $\vec{u}_T = \vec{F}_T / \|\vec{F}_T\|$;
- if $\|\vec{F}_T\| < k_f \|\vec{F}_N\|$, there is *non-sliding* contact, the point remains motionless, $\vec{F}_s = 0$.

k_f is called the *friction coefficient* ($k_f \in \mathbb{R}^+$). Note that if $k_f = 0$, there is sliding with *no friction*, and if $k_f = \infty$, there is necessarily no sliding at all. This coefficient is characteristic of the fabric's friction behavior. It has to be specified with its other characteristics (stiffness, elongation rate, it etc.).

In our model, these macroscopic laws of contact are adapted to the situation of collisions. They can indeed not be applied as such in a straightforward way, since the situation of contact occurs during an *infinitely small* time interval. This situation is at the limit of validity of Coulombian friction laws.

Consider a "point-triangle" collision where the triangle is motionless. The force generated by the impact of the point on the triangle (and *vice-versa*) is an unknown. Only the velocity $\vec{v}$ of the point *before the shock* is known. If $\vec{v}'$ is its velocity *after the shock*, then the acceleration of the point during $[t_0, t_0 + \Delta t]$ could approximated to $(\vec{v}' - \vec{v})/\Delta t$, and the force applied to this point by the triangle to $\vec{F}_c = \mu(\vec{v}' - \vec{v})/\Delta t$. But the thing is that $\vec{v}'$ is precisely what we have to determine, and is therefore also an unknown.

In order to solve this problem, we need to make an approximation so that the force generated by the impact can be evaluated. This approximation consists in considering that the forces implied are proportional to velocities, since it is obvious that the greater the impact velocity, the greater the force generated.

[2] We will consider in this section that the collision is *perfectly inelastic*, *i.e.* that there is no "bouncing" effect.

Whatever the coefficient of proportionality, the laws of friction described above can then be exactly rewritten by replacing $\overrightarrow{F}$ with $\overrightarrow{v}$ and $\overrightarrow{F}_s$ by $\overrightarrow{v}'$.

These relations therefore give us the velocity of the point *right after collision* against the triangle. Since the velocity of the point should be constant during $[t_0, t_0 + \Delta t]$, the algorithm simply replaces the velocity $\overrightarrow{v}$ of the point during the interval by $\overrightarrow{v}'$ and computes the corresponding trajectory from $P(t_0)$ to $P(t_0 + \Delta t)$. This is actually equivalent to considering that the collision precisely takes place at t_0, whatever the collision time $t \in [t_0, t_0 + \Delta t]$ that had been computed in the collision detection process.

3.2 Impact and dissipation

Another phenomenon during a collision is *impact* (in opposition to contact) and the collateral "bouncing" effect. During an "elastic" collision, there is no dissipation of energy at all. During an "inelastic" collision, there is such a dissipation, and a "perfectly inelastic" collision is a collision where the entire energy is dissipated.

This can be expressed with simple empirical relations. With the same notations as before, the velocity of the point P colliding the motionless triangle becomes after the shock: $\overrightarrow{v}' = \overrightarrow{v}_T - k_d \overrightarrow{v}_N$, where k_d is the dissipation coefficient ($0 \leq k_d \leq 1$). This coefficient is also part of the mechanical characteristics of the fabric.

3.3 Total response

In the general case, the velocity $\overrightarrow{v} = \overrightarrow{v}_T + \overrightarrow{v}_N$ of a point P before its collision with a motionless object therefore becomes after the collision:

$$\begin{cases} \text{If } \|\overrightarrow{v}_T\| \geq k_f \|\overrightarrow{v}_N\|, & \overrightarrow{v}' = \overrightarrow{v}_T - k_f \|\overrightarrow{v}_N\| \dfrac{\overrightarrow{v}_T}{\|\overrightarrow{v}_T\|} - k_d \overrightarrow{v}_N \\ \text{If } \|\overrightarrow{v}_T\| < k_f \|\overrightarrow{v}_N\|, & \overrightarrow{v}' = -k_d \overrightarrow{v}_N \end{cases} \tag{4}$$

In the case of moving objects, these relations are only applied to velocities once computed *in a reference frame moving at the velocity of the center of mass* of the object. In the case of self-collisions, the velocities are computed in a reference frame moving at the velocity of the center of mass of all elements involved in the collision (the point and the triangle, or the two edges).

In order to decompose the initial velocity of each mass point between its normal and tangential components, the normal used is the normal of the triangle at time t_0 in the case of a "point-triangle" collision. In the case of an "edge-edge" collision, the normal chosen is the result of the vectorial product of the two edges.

4 Consistency of multiple collisions

4.1 Multiple collisions

The collision handling algorithm presented so far is in fact insufficient for avoiding every case of self-penetration during a cloth animation. This collision algorithm indeed tackles only the problem of collisions between a *couple* of two elements, point and triangle, or edge and edge. These collisions are handled independently from each other, whereas in fact more than two of these elements may interfere with each other during a collision, over a time-step $[t_0, t_0 + \Delta t]$. Each computation of a collision modifies the positions of the points it involves, and it *also* modifies therefore the position of *all* the triangles and edges linked to these points, not only the ones directly involved in the collision. But nothing guarantees that these modifications did not create any other unpredicted collisions ... If this is the case, we will say that there are *multiple collisions*.

An interesting method for maintaining collision consistency can be found in [VMT95]. Our alternative method is based on the determination of the zones where these multiple collisions appear, and on handling their collisions specifically, with a new hypothesis of collision.

4.2 Determination of a "zone of impact"

The collision handling algorithm basically involves the position of the cloth model at time t_0 and its position at $t_0 + \Delta t$. Once collisions are handled, the computation of collision response has altered the position of the cloth at time $t_0 + \Delta t$.

In order to find out whether this computation has created new collision situations, a first thing to do can be to carry out one more collision detection. If the result of this detection is that there is no new collision, the algorithm can switch to the next time-step. If on the contrary new cases of collision appeared, it could also be possible to carry out another collision response computation, and then iterate. However, this iterative method is not guaranteed to converge.

The first phase of our method will be therefore to circumscribe all the points of the mesh that are involved in multiple interfering collisions. The iterative method described above will be used, with the aim to memorize at each iteration the set of points that are "linked", either because they take part in a *same* "point-triangle" or "edge-edge" collision, or because they take part in *two* different collisions that involve one or more points in common.

At each iteration, these sets of points, that we will name *zones of impact*, are likely to grow when new collision situations appear (see figure 3). During this growing process, if two zones of impact happen to include one or more points in common, *they are merged* so that they form a single larger zone of impact. The iterative method stops when all zones of impact stop to grow and remain stable: they are circumscribed.

This time, this method converges, since zones where multiple collisions occur are generally *local*. Moreover, even if it is not the case, zones of impact can

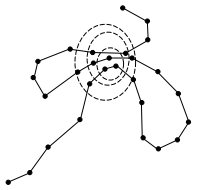


Figure 3: Iterative circumscription of zone of impact (cross-section view).

only grow to the point that they all have merged and have eventually included the whole cloth mesh. This ensures that there will never be an infinite loop.

4.3 Hypothesis of perfectly inelastic impact and non-sliding contact

The mere determination of zones of impact does not solve the handling of multiple collisions. The idea is to handle this zone of impact in a specific way so that no interpenetration occurs within the zone.

Note first that collisions occur between the time t_0 and the time $t_0 + \Delta t$, and it is guaranteed that there is no interpenetration yet at time t_0. Also, zones of impact are made of different elements of the mesh that interfere through multiple collisions: in a certain way, their movement is made difficult by these collision interferences, since they are all in contact with each other.

The idea is then to consider that all these imbricated elements will not be able to move but "as a whole", that is to say while remaining fixed with respect to each other, so that no collision occurrence may appear within the zone. This hypothesis of displacement is actually equivalent to suppose that within the zone, collision response consists in a *perfectly inelastic impact* and *non-sliding contact*. This is justified by the fact that, movement being made difficult by collision interferences, there is no possibility of any bouncing or gliding inside the zone.

Zones of impact eventually act as *rigid objects* during time-step $[t_0, t_0 + \Delta t]$. Their displacement is characterized by a *group velocity* $\overrightarrow{V}_G$ and a *group angular velocity* $\overrightarrow{\Omega}_G$.

$\overrightarrow{V}_G$ is computed as the mean velocity[3] of the n points of the zone of impact $\mathcal{Z}_c$:

$$\overrightarrow{V}_G = \frac{1}{n} \sum_{M \in \mathcal{Z}_c} \overrightarrow{V}_M$$

$\overrightarrow{\Omega}_G$ is computed for instance by reference to the geometric center G of

[3]For simplicity, points are all supposed to have the same mass; if it is not the case, then group velocities should be computed by reference to the center of mass.

$\mathcal{Z}_c$:

$$\begin{cases} \overrightarrow{OG} = \dfrac{1}{n} \displaystyle\sum_{M \in \mathcal{Z}_c} \overrightarrow{OM} \\ \overrightarrow{\Omega}_G = \dfrac{1}{n} \displaystyle\sum_{M \in \mathcal{Z}_c} \dfrac{\overrightarrow{GM} \wedge (\overrightarrow{V}_M - \overrightarrow{V}_G)}{\|\overrightarrow{GM}\|^2} \end{cases}$$

The collision response for all M in $\mathcal{Z}_c$ is therefore given by its new velocity:

$$\forall M \in \mathcal{Z}_c, \overrightarrow{V}'_M = \overrightarrow{V}_G + \overrightarrow{\Omega}_G \wedge \overrightarrow{GM}$$

4.4 Iteration

The hypothesis described above guarantees that no interpenetration will occur within each zone of impact during $[t_0, t_0 + \Delta t]$. However, to be accurate, nothing guarantees that the computation of their displacement do not create new collision occurrences at their *boundaries*. In order to be completely sure that this will not happen, the iterative circumscription of zones of impact must be coupled with the specific computation of collision response within these zones. The algorithm can be therefore divided in three phases.

1. The initial phase consists in detecting "point-triangle" and "edge-edge" collisions and computing their response without taking into account zones of impact.

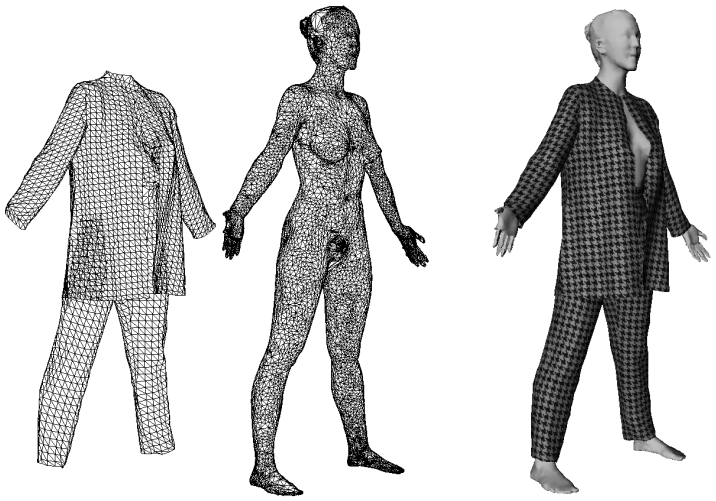
2. The second phase consists first in carrying out another collision detection and memorizing zones of impact if new collisions appeared. Then, a specific collision response of detected zones of impact is computed.

3. The third phase consists in iterating the second phase, making zones of impact grow or merge if necessary, until no new collision is detected.

This time again, this iterative method converges since zones of impact are most of the time local. If it were not the case (for instance if the cloth were all rumpled and rolled in a ball), all zones of impact would merge and eventually include the whole cloth, convergence would be however guaranteed.

Note that this iterative method takes place at each time-step. Once zones of contact have been successfully circumscribed and collision response has been fully solved, any memory of these zones of contact is erased, and they have to be computed again at the next time-step. It may then happen that the cloth object evolves in a way that new forces tend to separate some parts of the cloth that were in a same zone of contact. These parts therefore no longer collide with each other. They will hereafter not be included in a same zone of contact, unless they collide again.

In practice, even in the severe collision case of the falling ribbon shown in section 4.4, the crumpled zones unfold smoothly once it is mechanically and dynamically possible for them to do so.



(a) Meshes (shirt: 6000 polygons – trousers: 3000 – mannequin: 26,000).

(b) Rendered mannequin with garments.

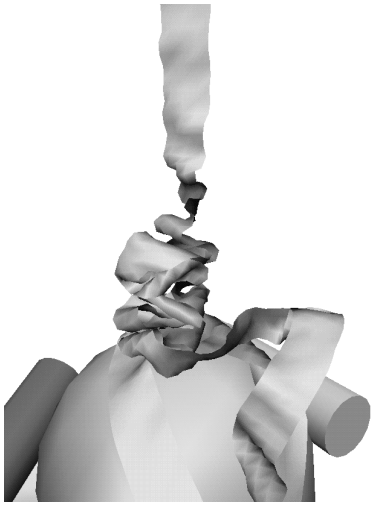
Figure 4: Garment construction.

Results and conclusion

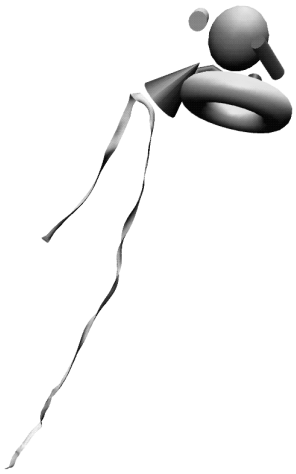
Our model has been used to simulate cases of cloth objects in various situations. The most achieved example is certainly the realization of garments to dress a virtual 3D mannequin. These garments are semi-automatically built using *real garment patterns* given by *Lectra Systèmes*, a specialized industry working in the field of computer-aided garment design. These patterns are then fit on a mannequin obtained from the scanning of a *real person*[4].

All collisions and self-collisions occurring during this garment construction have been successfully detected and handled, and no interpenetration of the clothes and the mannequin took place. The computation required for the building of clothes such as the shirt or the trousers shown in figure 4 took between one hour and a half to two hours for each, on a SGI Indigo 2. This is still an important computation time, but it is right away lower than the time required by a cloth modelist to build the real garment on a mannequin. Recent optimizations of the algorithm, not implemented at the time of the results presented above, allowed to decrease this computation time by 50 %. They were also tested in the critical case of a long ribbon colliding with a table and rumpling severely. The computation took one hour for six seconds of animation on a Dec Alpha 500/500.

[4]The automatic cutting of patterns and fitting on the 3D mannequin will be described in detail in my Ph.D. report [Pro97].



(a) Rumpled zone.



(b) Unfolded ribbon after the collision.

Figure 5: Falling ribbon (4000 polygons).

Acknowledgements

I would like to thank Georges Stamon and André Gagalowicz for their help during all our work, and Jean Marc Surville, from *Lectra Systèmes* (Bordeaux, France), who provided the garment patterns, and gave me many explanations and much advice. I also wish to thank the whole team of the *Projet Syntim* at INRIA, where this work has been carried out.

References

[Bar90] David Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *Computer Graphics (SIGGRAPH'90 proceedings)*, 24(4):19–28, août 1990.

[CYMTT92] Michel Carignan, Ying Yang, Nadia Magnenat-Thalmann, and Daniel Thalmann. Dressing animated synthetic actors with complex deformable clothes. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH'92 proceedings)*, volume 26, pages 99–104, juillet 1992.

[LKC96] J. D. Liu, M. T. Ko, and R. C. Chang. Collision avoidance in cloth animation. *The Visual Computer*, 12(5):234–243, 1996. ISSN 0178-2789.

[LMTT91] Benoit Lafleur, Nadia Magnenat-Thalmann, and Daniel Thalmann. Cloth animation with self-collision detection. In *Proc. of Conference on Modeling in Computer Graphics*. Springer, 1991.

[MW88] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. *Computer Graphics (SIGGRAPH'88 proceedings)*, 22(4):289–298, août 1988.

[Pro95] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface '95*, Québec, Canada, 17-19 mai 1995.

[Pro97] Xavier Provot. *Animation Réaliste de Vêtements.* PhD thesis, Université de Paris 5, printemps 1997. (to appear).

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Computer Graphics (SIGGRAPH'87 proceedings)*, volume 21, pages 205–214, juillet 87.

[VCMT95] Pascal Volino, Martin Courchesne, and Nadia Magnenat-Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH'95 proceedings)*, volume 29, pages 137–144, août 1995.

[VMT94] Pascal Volino and Nadia Magnenat-Thalmann. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. In *Computer Graphics Forum (EuroGraphics Proc.)*, volume 13, pages 155–166, 1994.

[VMT95] Pascal Volino and Nadia Magnenat-Thalmann. Collision and self-collision detection: efficient and robust solutions for highly deformable surfaces. In *6th Eurographics Workshop on Animation and Simulation*, pages 55–65, Maastricht, septembre 1995.