Accurate Collision Response on Polygonal Meshes

Pascal VOLINO, Nadia MAGNENAT THALMANN
MIRALab, C.U.I., University of Geneva - CH-1211, Switzerland
Email : [pascal|thalmann]@cui.unige.ch
Phone: -41 (22) 705 77 63 Fax: -41 (22) 705 77 80

Abstract

We present a very general geometrical correction method for enforcing collisions and other geometrical constraints between polygonal mesh surfaces. It is based on a global resolution scheme that takes advantage of an efficient use of the conjugate gradient algorithm to find the appropriate displacement of the mesh vertices that would satisfy all the constraints simultaneously, and according to momentum conservation laws. This method has been implemented in a cloth simulation system along with a collision response model that enforces a minimum "thickness" distance between cloth surfaces, which can be efficiently integrated in an simulation scheme based on implicit integration. Some provided examples illustrate the efficiency of the method.

1. Introduction

In simulation of dynamic systems, collision response is an intermediate task which consists in integrating to the mechanical model the effect produced by geometrical contacts between the objects of the system. A typical example illustrating the context of this paper is cloth simulation and garment animation, where cloth objects are dynamically simulated surfaces, for which contacts between each surface regions or with external objects should be integrated in order to relate reaction and friction effects.

Most of the time, deformable surfaces are represented by polygonal meshes, which is also the basic representation on which mechanical simulation will be performed. Polygonal meshes may also describe the surfaces of the other objects of the scene that can collide with the simulated surfaces. Hence, our problem is to find an efficient and accurate scheme to handle the collision response between objects described by polygonal meshes.

The most obvious way to consider and integrate collision response in a mechanical model is to build a model of the collision force, which illustrates reaction (normal component) and friction (tangential component). This approach is for instance considered in [LAF 91]. However, due to the discontinuous nature of geometrical contacts, this approach suffers from highly discontinuous and nonlinear force models, which perturb the numerical resolution of the mechanical equations. On the other hand, a geometric collision response model acting directly on the geometric state of the objects may be more efficient, as for instance presented in [EBE 96].

[VOL 95] and [VOL 97] presented a collision response scheme based on direct position, speed, and acceleration correction on the particles that are the vertices of the polygonal mesh. This scheme reproduced the effect of collisions without altering the mechanical simulation itself, because it did not consider additional contact reaction and friction forces which, being highly local and nonlinear, would perturb the numerical resolution of the simulation. Instead, the collisions were directly modeled through the reproduction of their expected final geometric and cinematic effects, in a very efficient manner.

However, several problems had to be considered: Contrary to mechanical collision forces, these geometrical corrections are not additive: When considering particles involved into several collisions, the state modification created by the enforcement of the constraints of one collision would modify the correction required for the other collisions. Simply adding up independently the corrections generated by all collisions is not an acceptable method. This is because complementary collisions would create an excessive correction, whereas antagonist collision corrections would cancel each other. This problem is particularly sensitive when dealing with simulations of deformable surfaces that can be stacked over several layers (Fig.1).

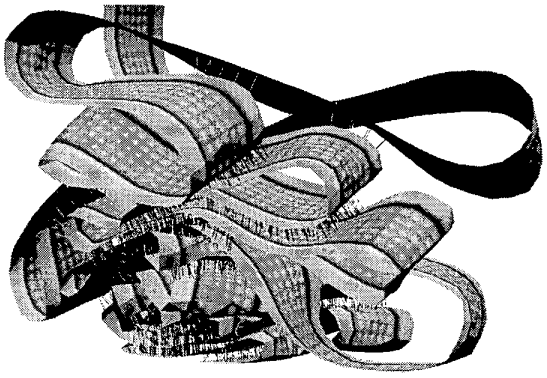

Fig.1: A highly complex collision network with numerous interacting collisions.

In [VOL 95] and [VOL 97], this problem was resolved by performing the corrections of each collision successively, after sorting the collision list from the most important to the least important needed correction: The correction of a given collision would be computed on the state already affected by the corrections of the previous collisions of the list. While this solution allowed the obtaining of acceptable results in most cases involving large collision regions or multi-layer collisions between several surfaces, this was mostly because the Midpoint and Runge-Kutta methods that were used for integrating the mechanical system numerically required the use of quite small time steps for computing the animation (down to 0.001 second in the "falling ribbon" presented in [VOL 95]). Also, iterating several times the collision list was sometimes necessary for scenes containing highly inter-dependant collisions. The resulting collision oversampling allowed the collision effects to get averaged to a stable configuration along time. However, through the more recent developments on efficient methods aimed to real-time systems, such as presented in [BAR 98], the large time-steps require highly accurate collision response, as errors would be amplified between the frames by the large time steps, and lead to inaccurate and unstable results incompatible with the smooth motions required for high quality animations.

This work intends to propose an accurate and efficient scheme for computing the corrections of the geometrical collision response approach, and keeping the benefits of simulating objects using simple mechanical models unperturbed by odd collision reaction and friction forces. We aim to be able to simulate highly complex scenes containing numerous interacting mechanically-simulated surfaces using efficient simulation methods such as the implicit method initially presented in [BAR 98].

Our proposed approach is a global resolution of all geometrical constraints generated by the collisions. Such global methods indeed already exist for many topics related to inverse kinematiks and managing contacts between rigid bodies, as discussed in [BAR 92], [BAR 96], [BRZ 88], [FUD 97], [SNY 93]. However, we intend to present a method specifically for managing collisions between polygonal meshes, that has to remain efficient if the number of collisions becomes large, and which can be easily integrated into efficient mechanical simulation schemes. The following points will be developed:

* In section 2, we present a rigorous approach for computing geometrical collision effects on polygonal meshes, ensuring basic mechanical conservation properties as well as good response continuity as the geometrical configuration of the collision evolves.

* In section 3, we detail a scheme that uses an efficient adaptation of the conjugate gradient method to enforce the constraints generated by the collision in a global way, taking into account simultaneously both mechanically simulated and geometrically constrained particles in a common way.

* In section 4, we describe a practical implementation of collision response in an implicit mechanical simulation system.

The paper is concluded in section 5 by examples showing the performances of the presented system.

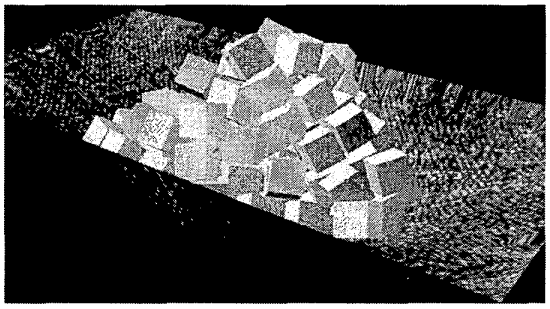

Fig.2: Complex interacting collisions.

2. Continuous Constraints in Polygonal Meshes

When dealing with polygonal meshes, usual collision detection algorithms return lists of colliding mesh features, which usually can either be vertices, edges, polygons. The collision response procedure has to interpret this data in terms of contact between the geometrical surfaces, and provide the adequate effect which, in the case of a particle system model, usually consists in exerting additional forces or altering the states of the mesh vertices involved in collisions. The particles, which are the vertices of the mesh, are the carriers of the geometrical information of the objects, and the effects of collisions have to be applied on them, in a suitable way satisfying dynamical correctness, response continuity, with the problem of dealing with particles involved in multiple collisions.

2.1. Collisions on Polygonal Meshes

Some simple cloth simulation applications would only consider a single simulated cloth interacting with external objects represented as non-mechanical surfaces. Collision response is fairly simple in these cases, as it merely consists in handling independently, for each vertex of the cloth mesh, the proximity to the non-mechanical object mesh (Fig.3). However, we intend to handle the very general situation where any mesh can be mechanically simulated, and all the forms of collisions that may occur between any of the elements of the mesh, possibly leading to mesh elements involved into several collisions.

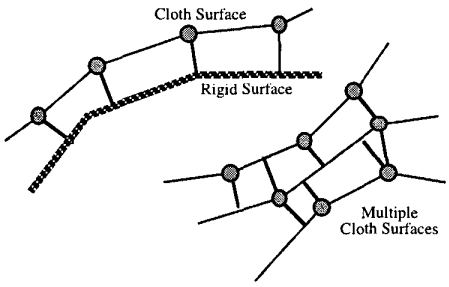

Fig.3: Collisions in polygonal meshes: Simple cloth-rigid approach, General approach.

Usually, collisions may be sorted out as *intersections*, where two surface elements interpenetrate each other, and *proximities*, where two surface elements are separated by a distance below a given threshold, usually representing the "thickness" of the surface. In the case of collisions between polygonal meshes, we can identify the following cases (Fig.4):

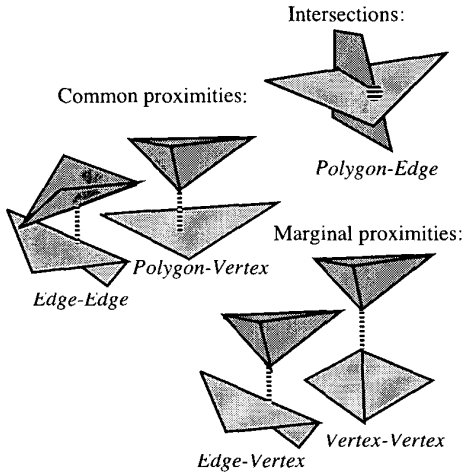

Fig.4: Intersections and proximities in polygonal meshes.

Collision response may either handle intersections usually by backtracking the motion leading to the surface crossing and integrating the collision effect, and proximities by maintaining a minimum separation distance between the surfaces (Fig.5). In either case, the collision effect is applied on the mesh vertices of the colliding elements.

A good collision response scheme has to exhibit continuity properties: A slight change in the state of the colliding elements should only produce a slight change in the collision response. This property is essential for producing high quality animations where the objects do not "jump" as they slide on each other.

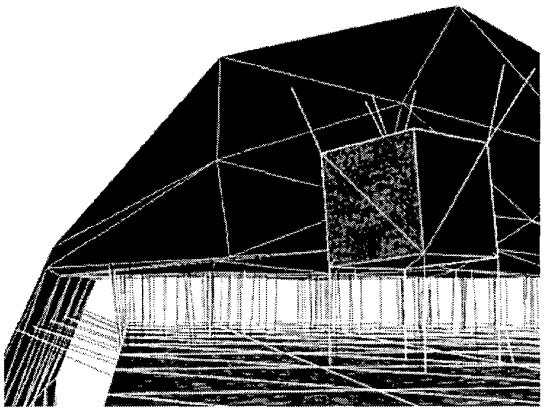

Fig.5: Different proximity kinds in polygonal mesh collisions.

2.2. Continuous Constraint Collision Response

In our general approach, a collision is exerted on a point of any mesh element (vertex, edge, polygon), rather than solely on a single vertex. The first aspect of our proposed scheme considers a collision **c** between two mesh elements **A** and **B**, each being either mesh vertices, edges or polygons identified by their respective vertex list $\mathbf{i} \in \mathbf{A}$. and $\mathbf{i} \in \mathbf{B}$. The collision is identified by its two contact points on the mesh elements **A** and **B**, and the *collision relative position* **Pc** being the difference of the contact point on object **A** minus the contact point on object **B**. Barycentric coordinates of the contact points positions on their respective mesh elements can be computed with respect to mesh vertex positions **Pi** with simple geometry. We turn the difference into a sum by calling **Sci** the (positive) barycentric coordinates of the contact point on mesh element **A** and the opposite (negative) of the barycentric coordinates on mesh element **B** (Fig.6). The collision relative position **Pc** then verifies the following relation:

$$\mathbf{Pc} = \sum_{i \in C} \mathbf{Sci}\ \mathbf{Pi} \qquad (1)$$

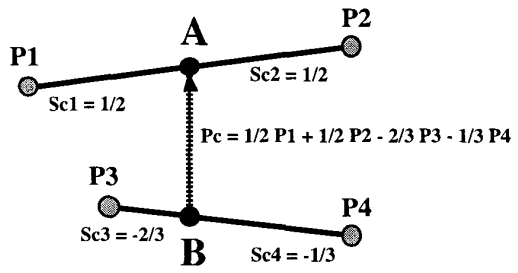

Fig.6: Example of collision between two edges **A** and **B**., and computation of the collision relative position **Pc**.

The Geometrical Property Correction Process

The geometric collision correction scheme aims to correct a geometrical property of the collision, that is either position, speed or acceleration, that is measured by the difference of this property between the two contact points, in the same way as for the collision relative position illustrated in Fig.1. First, for all collision **c**, we compute a geometric property of the collision **Qc** that is either relative position **Pc**, velocity **Pc'**, acceleration **Pc"**, by interpolating the corresponding property of the particles **Qi** (Fig.7.1) using the barycentric coordinates similarly as in (1), as follows:

$$\mathbf{Qc} = \sum_{i \in C} \mathbf{Sci}\ \mathbf{Qi} \qquad (2)$$

From this, a collision correction model will compute the expected geometrical correction $\Delta_0\mathbf{Qc}$ (either relative position, speed, acceleration) to be exerted by the collision from it (Fig.7.2), as will be detailed in section 4.1. We should then compute for every collision **c** the individual alterations $\delta\mathbf{Qc}$ (Fig.7.3) that will modify the property of the particles **i** involved in the collision by an amount

$\delta\mathbf{Qci}$ using a distribution that fulfills mechanical conservation laws according to the collision geometry and particle weights (Fig.7.4), as discussed next. We finally expect that the combined effect $\Delta\mathbf{Qi}$ of all these corrections on the state $\Delta\mathbf{Qc}$ of any collision **c** to be equal to the expected correction $\Delta_0\mathbf{Qc}$ for all collisions simultaneously (Fig.7.5), as discussed in section 3. The geometrical properties of the particles are finally corrected using the computed values (Fig.7.6), and the collisions then satisfy the collision response model.

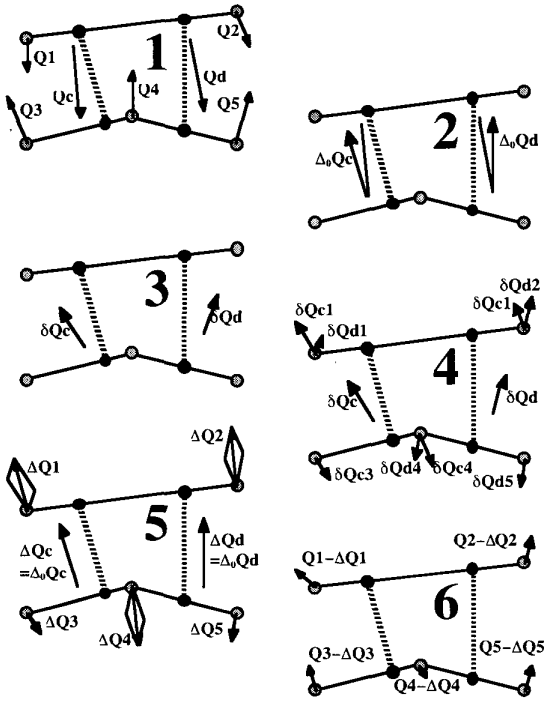

Fig.7: Successive steps of the geometrical corrections of the collision response process. Example with two interacting collisions **c** and **d**.

Distributing Correction in the Particles

The applied response correction $\delta\mathbf{Qc}$ resulting from the collision **c** can be represented as a sum of contributions $\delta\mathbf{Qci}$ for all particle **i** involved in the collision, as follows:

$$\delta\mathbf{Qc} = \sum_{i \in C} \mathbf{Sci}\ \delta\mathbf{Qci} \qquad (3)$$

We should then compute the $\delta\mathbf{Qci}$ contributions individually by distributing the correction $\delta\mathbf{Qc}$ according to momentum conservation laws involving the consideration of the particle masses that we call $\mathbf{Mi}$ (Fig.8).

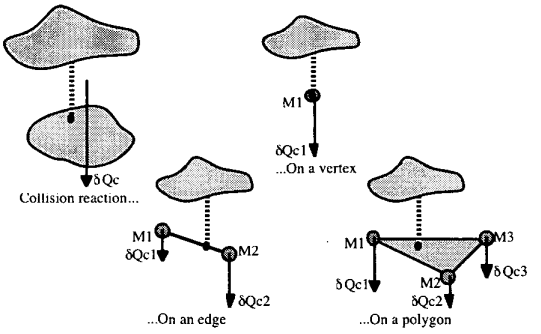

Fig.8: Repartition of collision response on the vertices of a mesh element.

If we model the correction $\delta\mathbf{Qc}$ to be the result of an interaction force $\mathbf{Fc}$, momentum conservation imposes that the forces $\mathbf{Fci}$ exerted on the individual particles is weighted by $\mathbf{Sci}$. By defining a proportionality coefficient γ equal to $\mathbf{1}$, $\mathbf{dt}$ or $\mathbf{dt^2/2}$ as $\mathbf{Q}$ represents acceleration, speed or position respectively, we then obtain:

$$\delta\mathbf{Qci} = \mathbf{Mi^{-1}}\ \gamma\ \mathbf{Fci} = \mathbf{Mi^{-1}}\ \mathbf{Sci}\ \gamma\ \mathbf{Fc} \qquad (4)$$

Using relation (3), the total desired correction $\delta\mathbf{Qc}$ resulting from the force $\mathbf{Fc}$ is then expressed as follows:

$$\delta\mathbf{Qc} = \sum_{i \in C} \mathbf{Sci}\ \mathbf{Mi^{-1}}\ \mathbf{Sci}\ \gamma\ \mathbf{Fc} \qquad (5)$$

Substituting $\gamma\ \mathbf{Fd}$ of relation (5) by its expression obtained from relation (4), we can finally express the correction to be exerted on each particle as follows:

$$\delta\mathbf{Qci} = \mathbf{Mi^{-1}}\ \mathbf{Sci}\left(\sum_{i \in C} \mathbf{Sci}\ \mathbf{Mi^{-1}}\ \mathbf{Sci}\right)^{-1}\ \delta\mathbf{Qc} \qquad (6)$$

We can easily check the continuity of this formula when the collision points move from one mesh element to another (some barycentric coordinates reach $\mathbf{0}$). We also verify that if the collision point is in the middle of any mesh elements (identical barycentric coordinates of all vertices of that element), all vertices of the element get an identical correction. For the commodity in the subsequent formulas, we define the "reduced mass" $\mathbf{Mc}$ of the collision **c** as follows:

$$\mathbf{Mc^{-1}} = \sum_{i \in C} \mathbf{Sci}\ \mathbf{Mi^{-1}}\ \mathbf{Sci} \qquad (7)$$

And, from expression (6), the correction to be exerted on each particle is:

$$\delta\mathbf{Qci} = \mathbf{Mi^{-1}}\ \mathbf{Sci}\ \mathbf{Mc}\ \delta\mathbf{Qc} \qquad (8)$$

Constrained Particles

It is quite simple to include constrained particles in this scheme, by considering that such particles do not react to external forces by having infinite mass. Hence, such particles can simply be handled by considering that $\mathbf{Mi^{-1}} = \mathbf{0}$ for them in the above formulas.

This scheme can even be generalized to particles constrained to move along particular directions by replacing in the above formulas the $\mathbf{Mi^{-1}}$ scalar values by hermitian inertia matrices having null eigenvalues for the

eigenvectors along the constrained directions, and eigenvalues equal to the inverse mass along unconstrained directions. For instance, for constraining a particle **i** of mass **mi** to move along the direction defined by the normalized **X** vector, we would use in the above formulas the matrix $\mathbf{Mi^{-1}} = \mathbf{mi^{-1}}\ \mathbf{X}\ \mathbf{X^T}$. For constraining it to move along the plane orthogonal to that direction, we would use the matrix $\mathbf{Mi^{-1}} = \mathbf{mi^{-1}}\ (\mathbf{I} - \mathbf{X}\ \mathbf{X^T})$. An unconstrained particle is also obviously handled with a scalar matrix $\mathbf{Mi^{-1}} = \mathbf{mi^{-1}}\ \mathbf{I}$. A similar scheme was proposed in [BAR 98] for constraining particle motion in their implicit simulation system.

A collision on an over-constrained particle system has a null inverse reduced mass ($\mathbf{Mc^{-1}} = \mathbf{0}$). In order to prevent this to cause trouble in the computation, we suggest adding to it a small constant value (or a small constant scalar matrix when dealing with inertia matrices) in order to remove the singularity that may arise for computing the reduced mass **Mc** in this case.

3. Efficient Global Constraint Resolution

The main problem with defining a collision response scheme based on geometrical corrections results from the fact that the collisions usually are not independent: Some particles of the system may be involved into several collisions simultaneously (Fig.9). Any geometrical correction performed by one collision on such a particle thus also modifies the geometrical state of all the other collisions in which it is involved. An efficient collision response scheme should compute the corrections $\Delta\mathbf{Qi}$ to be applied on the particles so as the resulting effective global correction $\Delta\mathbf{Qc}$ becomes equal to the desired correction $\Delta_0\mathbf{Qc}$ for all collision **c** simultaneously.

Because of the dependency of the collisions on each other, we cannot simply add the collisions corrections independently. The result of this naive approach would be an over-correction on mesh elements containing several complementary collisions (multiple collisions between two surfaces colliding in a large area), and an under-correction on mesh elements containing antagonist collisions (collisions between several surfaces stacked on multiple layers) (Fig.9). Solving this problem is however necessary for obtaining an acceptable collision response scheme.

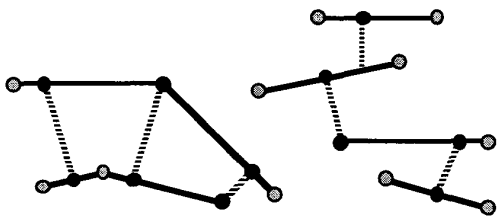

Fig.9: Interacting collisions: Complementary, and .antagonist collisions.

3.1. The Successive Correction Scheme

The collision response algorithm described in [VOL 97] uses the following algorithm:

For each collision c

Compute $\mathbf{Qc} = \sum_{i \in C} \mathbf{Sci}\ \mathbf{Qc}$

Compute $\Delta_0\mathbf{Qc}$ **from Qc**

with the collision response scheme

Add $\delta\mathbf{Qci} = \mathbf{Mi^{-1}}\ \mathbf{Sci}\ \mathbf{Mc}\ \Delta_0\mathbf{Qc}$ **to Qi**

for each particle i involved in c

This scheme handles all the collisions successively, applying a collision correction $\delta\mathbf{Qc}$ equal to the desired correction $\Delta_0\mathbf{Qc}$ on a system state already affected by the corrections of former collisions. This is the simplest way to deal with the problem of complementary collisions which act on common particles in the same direction, a situation which happens quite frequently when large portions of surfaces collide. However, this algorithm shows its deficiency when it has to deal with large networks of antagonist collisions, which for example occur when many surfaces are stacked on each others, and where the correction of a collision partially cancels the previously applied corrections of antagonist collisions. The algorithm implemented in [VOL 97] improved the situation by sorting collisions from the smallest to the largest proximity distance, and iterated the corrections several times in difficult situations. While it still would produce good results in complex situations when small time steps would allow the collisions to stabilize over several iterations, new algorithm require a much more accurate approach which is presented in the following section.

3.2. An Accurate Global Correction Scheme

Our new algorithm intends to compute the collision corrections on each particle in a global way that should simultaneously satisfy to the constraints defined by all collisions.

Building the System

Our idea is to solve for all collisions the equality between the effective and desired collision corrections as follows:

$$\Delta\mathbf{Qc} = \Delta_0\mathbf{Qc} \qquad (9)$$

The effective correction of a collision **c** results from the total corrections $\Delta\mathbf{Qi}$ on the vertices, and is expressed similarly as (2) as follows:

$$\Delta\mathbf{Qc} = \sum_{i \in C} \mathbf{Sci}\ \Delta\mathbf{Qi} \qquad (10)$$

Combining (9) and (10), we have:

$$\sum_{i \in C} \mathbf{Sci}\ \Delta\mathbf{Qi} = \Delta_0\mathbf{Qc} \qquad (11)$$

The total vertex corrections $\Delta\mathbf{Qi}$ result from the combined effect sum of responses $\delta\mathbf{Qdi}$ computed with formula (8) from the corrections $\delta\mathbf{Qd}$ for all collisions **d** which involve the vertex **i** of the scene, as follows:

$$\Delta\mathbf{Qi} = \sum_{d \ni i} \delta\mathbf{Qdi} = \sum_{d \ni i} \mathbf{Mi^{-1}}\ \mathbf{Sdi}\ \mathbf{Md}\ \delta\mathbf{Qd} \qquad (12)$$

Combining equation (11) and formula (12), we express how all the individual collision corrections $\delta\mathbf{Qd}$ have to combine to obtain the desired correction $\Delta_0\mathbf{Qc}$, as follows:

$$\sum_{i \in C} \mathbf{Sci} \sum_{d \ni i} \mathbf{Mi^{-1}\ Sdi\ Md\ \delta Qd} = \Delta_0\mathbf{Qc} \qquad (13)$$

After rearrangements, we get the final linear equation system defined for all collision **c**:

$$\sum_{d} \left(\sum_{i \in C \cap D} \mathbf{Sci\ Mi^{-1}\ Sdi} \right) (\mathbf{Md\ \delta Qd}) = \Delta_0\mathbf{Qc} \qquad (14)$$

This linear system is symmetric, and allows efficient implementation of numerical resolution methods, discussed as follows. We also check that if the collisions are all independent (if each vertex **i** belongs to no more than one collision) and considering expression (7), we obtain $\delta\mathbf{Qc} = \Delta_0\mathbf{Qc}$.

3.3. Resolution of the Linear System

We now have to solve a linear system $\mathbf{H\ X = Y}$. The linear system is symmetric, and usually sparse considering that each collision only interacts with a small number of other collisions. Due to the structure of the matrix elements, it is also positive (no negative eigenvalues). However, it is not likely to be definite, mostly because the collisions do not constrain the particles totally (under-constrained system). Thus, we need a method which is able to deal with degenerate equation systems for "ignoring" the degrees of freedom irrelevant to the desired solution.

The Conjugate Gradient method, well-known for the efficient resolution of sparse symmetric definite-positive linear systems, appears to be the best resolution scheme in our case, as it is an iterative method which converges quite quickly to a solution through a "relaxation" scheme which is perfectly symmetric to all equations and all variables.

Linear system resolutions are traditionally avoided, because of the hassle they imply for their resolution. Sparse systems are usually associated to dedicated complicated data structures, leading to complex algorithms for their construction and processing. Furthermore, considering a particle involved in **n** different collisions, the number of terms added by that particle to the **H** matrix is $\mathbf{n^2}$. We will however see that for our problem, these complications can be avoided through an adequate implementation of the Conjugate Gradient algorithm which does not require explicit representation of the system matrix **H**.

Integration into the Conjugate Gradient Method

Using **d** for indexing the unknowns and **c** for indexing the equations of relation (14) for collisions, the equation members are expressed as follows:

$$\mathbf{Hcd} = \sum_{i \in C \cap D} \mathbf{Sci\ Mi^{-1}\ Sdi} \qquad \mathbf{Yc} = \Delta_0\mathbf{Qc} \qquad \Delta_*\mathbf{Qd} = \mathbf{Md^{-1}\ Xd} \qquad (15)$$

With α, β as intermediate scalars and **R**, **T** as intermediate vectors, the Conjugate Gradient algorithm can be expressed as follows:

$\beta \leftarrow \mathbf{0}$; $\mathbf{X} \leftarrow \mathbf{0}$; $\mathbf{R} \leftarrow \mathbf{Y - H\ X}$

$\alpha \leftarrow \mathbf{R^T\ R}$; **if** $(\beta \neq \mathbf{0})$ $\mathbf{T} \leftarrow \mathbf{R} + (\alpha/\beta)\ \mathbf{T}$ **else** $\mathbf{T} \leftarrow \mathbf{R}$

$\beta \leftarrow \mathbf{T^T\ H\ T}$; $\mathbf{R} \leftarrow \mathbf{R} - (\alpha/\beta)\ \mathbf{H\ T}$; $\mathbf{X} \leftarrow \mathbf{X} + (\alpha/\beta)\ \mathbf{T}$

$\beta \leftarrow \alpha$

until adequate convergence

In this algorithm, the **H** matrix does only participate in products with the vector **T**. Thanks to the appropriate structure of the matrix **H**, the resulting terms can be computed easily. From expressions (15), we obtain:

$$(\mathbf{H\ T})_c = \sum_{d} \mathbf{Hcd\ Td} = \sum_{i \in C} \mathbf{Sci\ Mi^{-1}} \left(\sum_{d \ni i} \mathbf{Sdi\ Td} \right) \qquad (16a)$$

$$(\mathbf{T^T\ H\ T}) = \sum_{c \times d} \mathbf{Tc\ Hcd\ Td} = \sum_{i} \left(\sum_{c \ni i} \mathbf{Tc\ Sci} \right) \mathbf{Mi^{-1}} \left(\sum_{d \ni i} \mathbf{Sdi\ Td} \right) \qquad (16b)$$

We can see that the sums within parentheses are not nested and can therefore all be evaluated independently for each vertex **i**. This is a key of an efficient implementation that would combine these two evaluations in a single procedure, and which becomes particularly simple in the case of having scalar **M** and **S** terms. Our implementation does not express explicitly the matrix **H** through any kind of complicated sparse matrix data structure, but simply computes expressions (16) on the fly within each iteration of the Conjugate Gradient algorithm. The only implementation difficulty is actually to design a data structure which allows, for any particle, to find efficiently all the collisions in which this particle is involved.

The New Resolution Scheme

We implement the resolution scheme as follows:

For each collision c

Compute $\mathbf{Qc} = \sum_{i \in C} \mathbf{Sci\ Qc}$

Compute $\mathbf{Yc} = \Delta_0\mathbf{Qc}$ **from Qc**

with the collision response scheme

Solve $\mathbf{H\ X = Y}$

For each collision d

Add $\delta\mathbf{Qdi} = \mathbf{Mi^{-1}\ Sdi\ Md\ \delta Qd} = \mathbf{Mi^{-1}\ Sdi\ Xd}$ **to Qi**

for each particle i involved in d

The convergence of the Conjugate Gradient algorithm can be evaluated with the value β, illustrating the residual quadratic error on collision constraints. However, we found it sufficient to iterate the algorithm only a fixed number of times, as convergence is usually quite fast. For instance, only one iteration solves most of the simple cases, and a few iterations also resolve complex interacting collisions. It is however important to avoid the numerical artifacts that may arise from limited number representation accuracy, which can be tracked by a sudden significant increase of β during the iterations.

Implementing Collision Response

The previous section has discussed the resolution of a set of geometrical constraints defined by collisions in a polygonal mesh. An example of adequate computation of these constraints from geometrical collision properties is now described in this part, along with an implementation of a mechanical model taking advantage as well of the previously discussed algorithms.

4.1. A Collision Correction Model

A geometrical correction scheme takes into account collision effects by altering the geometrical state of the system. From the current state, the system is corrected to its desired state corresponding to what is expected with the current collisions. The most obvious factor is the avoidance of geometrical interferences between the surfaces, that should be ensured and maintained along time.

Our geometrical collisions response scheme consists in correcting the positions ($\mathbf{Q} \leftarrow \mathbf{P}$) so as to prevent surface crossing, and correcting speed ($\mathbf{Q} \leftarrow \mathbf{P'}$) and acceleration ($\mathbf{Q} \leftarrow \mathbf{P''}$) so as to cancel out normal components pushing the mesh elements toward each other. Additional friction effects were obtained using corrections on tangential components. These corrections should be applied after the computation of the force and acceleration $\mathbf{P''}$ out of the system state $\mathbf{P}$ and $\mathbf{P'}$ as defined by the mechanical properties, and before the numerical integration that will compute from it the system state at the next time step. This "triple correction" on the position, speed and acceleration components allow the modeling of collision response to be as "passive" as possible, always based on an attenuation of relative collision speed and acceleration, and thus energy dissipation guarantying stability whatever the timestep used for the simulation.

Our implementation intends to simulate surface thickness by maintaining the geometrical meshes of different surface regions at a given distance from each other. The corrections are applied successively as follows (Fig.10):

* The **position correction** alters the position $\mathbf{P}$ of the colliding vertices so that the collision distance is maintained at the current frame. Given the wanted collision position $\mathbf{Pc_0(t)}$ at the current frame, the correction should be:

$$\Delta_0 \mathbf{Pc(t)} = \mathbf{Pc_0(t)} - \mathbf{Pc(t)} \qquad (17)$$

* The **speed correction** alters the speed $\mathbf{P'}$ of the colliding vertices so that the collision distance is obtained at the next frame. Given the desired position $\mathbf{Pc_0(t+dt)}$ at next frame, the speed correction should be:

$$\Delta_0 \mathbf{Pc'(t)} = \frac{\mathbf{Pc_0(t+dt)} - \mathbf{Pc(t)}}{\mathbf{dt}} - \mathbf{Pc'(t)} \qquad (18)$$

* The **acceleration correction** alters the acceleration $\mathbf{P''}$ of the colliding vertices so that the collision distance is obtained two frames thereafter with null distance evolution. Given the wanted position $\mathbf{Pc_0(t+2dt)}$ and (null) speed $\mathbf{Pc_0'(t+2dt)}$ at second next frame, the acceleration correction should be:

$$\Delta_0 \mathbf{Pc''(t)} = \frac{\mathbf{Pc_0(t+2\,dt)} - \mathbf{Pc(t)}}{\mathbf{dt^2}} - \frac{\mathbf{0.5\,Pc_0'(t+2\,dt) + 1.5\,Pc'(t)}}{\mathbf{dt}} - \mathbf{Pc''(t)} \qquad (19)$$

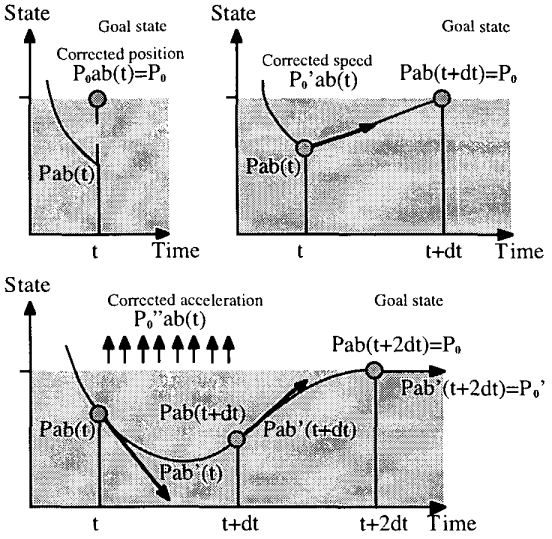

Fig.10: Position, speed and acceleration correction schemes.

Depending on the simulation context, the use of these three successive schemes appears to be the most robust solution, using specific collision distances for each one. Practically, position correction is first applied in order to replace the particles in "reasonable" configurations, speed correction then orients the particle evolution toward a better configuration, and acceleration correction finally ensures smooth evolution toward the desired configuration (Fig.11). Collision response should in practice rely mostly on the latter, whereas the two others act depending on the collision severity, typically during initial shock.

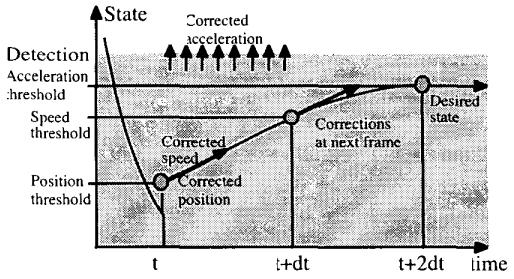

Fig.11: Combined corrections of position, speed and acceleration.

Collision reaction is simulated by performing these corrections on the normal component of the collision according to the collision orientation. Coulombian (solid) friction is simulated by altering the tangential components, and reducing their amplitude by a tangential correction component for which the norm is the norm of the normal correction multiplied by the specified friction coefficient, and total suppression if this is higher than the initial tangential value (non-sliding collision). This correction should mainly be applied for acceleration, but may also concern speed and position.

4.2. Integration in an Implicit Simulation Scheme

The recent advances in cloth simulations are based on implicit numerical integration schemes, as described in [BAR 98]. Our system uses an adapted variation of this system described in [VOL 00], which uses a simplified implementation of the Conjugate Gradient algorithm that do not handle explicitly the force derivative sparse matrix, in a way very similar to what is described in Part.3.3.

Implicit models perform very well with mechanical forces that can be computed analytically, and for which the derivatives on the system state can be computed. It is difficult to integrate geometrical collision response schemes, as they are not based on mechanical forces.

Fortunately, the acceleration correction described in the previous part may indeed be considered as the result of "collision forces", which can be determined analytically from the system state and the corrections that are computed from it. Considering a collision response scheme that mostly relies on acceleration correction, full benefit of implicit integration can be restored through adequate contribution of the collision effects in the force derivative matrix.

The inverse Euler step computes the evolution of the system using the iteration:

$$\mathbf{P}'(\mathbf{t}+\mathbf{dt}) = \mathbf{P}'(\mathbf{t}) + \mathbf{H}^{-1}\,\mathbf{Y}$$
$$\mathbf{P}(\mathbf{t}+\mathbf{dt}) = \mathbf{P}(\mathbf{t}) + \mathbf{P}'(\mathbf{t}+\mathbf{dt})\,\mathbf{dt} \qquad (20)$$

Calling **M** the (diagonal) mass matrix of all the particles of the system and **F** the force vector exerted on all the particles, the components of the linear system to be resolved are:

$$\mathbf{H} = \mathbf{M} - \frac{\partial \mathbf{F}}{\partial \mathbf{P}'}\,\mathbf{dt} - \frac{\partial \mathbf{F}}{\partial \mathbf{P}}\,\mathbf{dt}^2$$
$$\mathbf{Y} = \mathbf{F}(\mathbf{t})\,\mathbf{dt} + \frac{\partial \mathbf{F}}{\partial \mathbf{P}}\mathbf{P}'(\mathbf{t})\,\mathbf{dt}^2 \qquad (21)$$

We compute the equivalent "force" **Fcj** exerted on the particle **j** by a collision **c** during the timestep **dt** using relations (4) and (8) as follows:

$$\mathbf{Fcj} = \mathbf{Scj}\,\mathbf{Mc}\,\Delta_0\mathbf{Pc}'' \qquad (22)$$

With relation (2), the force partial derivatives on position or speed evolution **Qi** of particle **i** are computed as follows:

$$\frac{\partial \mathbf{Fcj}}{\partial \mathbf{Qi}} = \mathbf{Scj}\,\mathbf{Mc}\,\mathbf{Sci}\,\frac{\partial \Delta_0\mathbf{Pc}''}{\partial \mathbf{Qc}} \qquad (23)$$

The right-hand partial derivatives are obtained from the acceleration response model, which in our case is expressed by relation (19), as follows:

$$\frac{\partial \Delta_0\mathbf{Pc}''}{\partial \mathbf{Pc}} = -\mathbf{dt}^{-2} \qquad \frac{\partial \Delta_0\mathbf{Pc}''}{\partial \mathbf{Pc}'} = -\mathbf{1.5}\,\mathbf{dt}^{-1} \qquad (24)$$

And, from (21), we obtain additional terms of the force derivative matrix, for a collision **c**:

$$\mathbf{Hji} = \mathbf{2.5}\,\mathbf{Scj}\,\mathbf{Mc}\,\mathbf{Sci} \qquad (25)$$

As discussed in Part.3.3, we implement these corrections directly in the Conjugate Gradient algorithm, through the computation of extra contributions in the products of the matrix with an arbitrary vector **H T** and $\mathbf{T}^T\,\mathbf{H}\,\mathbf{T}$. Considering the additive effects of all collisions, the extra values should be added:

$$(\mathbf{H}\,\mathbf{T})_j = \sum_i \mathbf{Hji}\,\mathbf{Ti} = \mathbf{2.5}\sum_{c\ni j}\mathbf{Scj}\,\mathbf{Mc}\left(\sum_{i\in c}\mathbf{Sci}\,\mathbf{Ti}\right) \qquad (26a)$$

$$(\mathbf{T}^T\,\mathbf{H}\,\mathbf{T}) = \sum_{j,i}\mathbf{Tj}\,\mathbf{Hji}\,\mathbf{Ti} = \mathbf{2.5}\sum_c\left(\sum_{j\in c}\mathbf{Tj}\,\mathbf{Scj}\right)\mathbf{Mc}\left(\sum_{i\in c}\mathbf{Sci}\,\mathbf{Ti}\right) \qquad (26b)$$

The computation of these terms is simple and straightforward. Our implementation considers a similar formulation for expressing the forces resulting from the mechanical model, and computing their derivatives, yielding efficient integration. It also performs a preconditioning by weighting equations and unknowns by the square root of the particle inverse masses, yielding good convergence even if the particles have very disparate masses, and suppressing the need of dealing with constrained particles as particular cases.

4.3. Collision Detection

Collision detection is performed using the hierarchical scheme described in [VOL 94]. This algorithm builds a hierarchy on the mesh polygons, and constructs a surface region tree as a preprocessing task. Collision detection is performed by testing collisions between the bounding volumes of the surface regions of the hierarchy. Self-collision is efficiently computed by using a surface curvature criteria describing the possibility of collisions within a given surface region, and that replace the bounding volume test when testing adjacent surface regions.

The algorithm was improved by replacing the bounding boxes used in [VOL 94] by Discrete Oriented bounding Polytopes as described in [KLO 97], choosing 12-sided volumes described by the face orientations of a dodecahedron.

5. Results

We have implemented this algorithm in an experimental cloth mechanical simulation system that takes advantages of the technologies described in part 4.2. We describe here two representative examples illustrating the possibilities of the presented algorithm.

5.1. The Monster Pile

In this test, we let fall 34 identical soft rectangles on a fixed base, horizontally. Each rectangle is composed of 132 triangles (Fig.12). The collision model has a high friction coefficient, preventing the rectangles to slide on each other.

The bottleneck for this computation is clearly the collision detection, which, when the pile gets complete, takes more than 80% of the total computation time. On an SGI Octane, the computation can be considered as "realtime" when the pile has less than five layers.

In this example, we can clearly see that collision distance is maintained between all the layers of the pile. While the time taken for handling collision response using the old [VOL 97] algorithm is approximately half than for the one presented here, it was not able to preserve distance and stability so well in all the layers of the stack. Our new

algorithm is able to propagate the collision constraints efficiently through all the layers of the pile, and maintain the collision constraints between each layer.

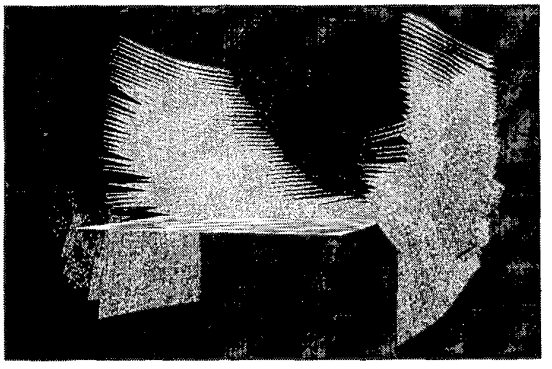

Fig.12: 34 soft rectangles piled on each other.

5.2. The New Ribbon

Similarly to a test presented in [VOL 95], we let a very long ribbon fall on a fixed ground. The ribbon is a polygonal mesh with approximately 11500 triangles (Fig.13). The simulation start with friction, which is after a moment suppressed to let the ribbon slide away from its support. The whole animation contains about 1600 frames.

At the difference of the scene presented in [VOL 95], the computation speed gets good benefit from our new algorithm. While the old system required several hundreds of iterations to be computed for each animation frames, only between five and ten iterations per step are required with the new method (this value is actually determined by the maximum displacement allowable for correct collision detection between each iteration). The collision response scheme can still handle collisions accurately, as shown in this example.

In terms of performance, collision detection accounts for roughly 50% of the total computation time (typical value in the middle of the simulation), while collision response accounts for 20% and mechanical computation 30%. One frame is typically computed in about 30 seconds on an SGI Octane.

6. Conclusion

The geometrical collision response method described in this work is an accurate and efficient, yet very general method for dealing with collisions and other geometrical constraints that may be included in mechanical simulation systems dealing with deformable polygonal meshes. This approach can for instance accurately propagate collision effects through several layers of objects, despite the large iteration timesteps used in efficient mechanical simulation methods.

While the implementation detailed in this paper enforces distance constraints between the objects, the scheme can be used in collision response models that use other forms of collision models, such as preventing object interference. We have also been able to define other kinds of constraints not directly related to collisions, such as "elastics" which smoothly pulls two arbitrary surface points together and maintains them attached. Hence, the presented approach can be adapted to many different applications which involve enforcing geometrical constraints.

There are still several issues to solve for perfecting the presented method. First, we would like to take into account the "inequality" nature of the collision constraints possibly using a modification in the Conjugate Gradient algorithm, in order to improve the resolution for certain cases of

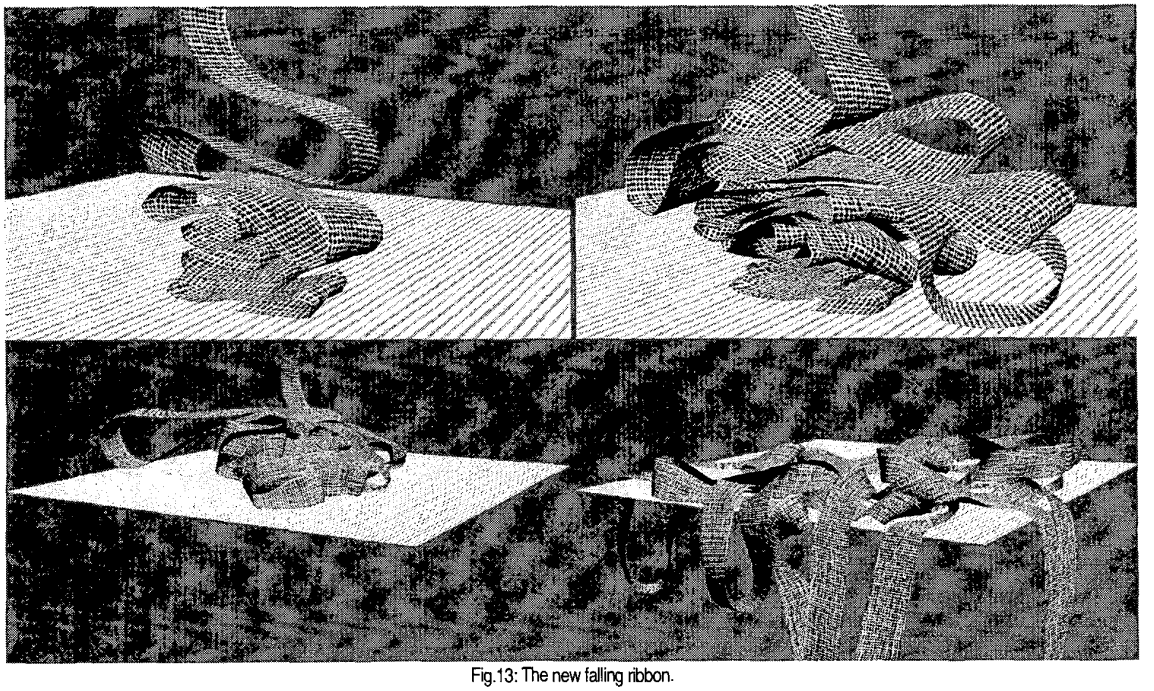

Fig.13: The new falling ribbon.

complementary interacting collisions. Then, we would like to improve smooth integration of collision response with the implicit Midpoint method, possibly using additional correction of the remnant system state the this method requires. There are also issues concerning stability of interacting collisions without frictions, where some possible configuration create instability, as discussed in part.3.4.

We are currently in process of integrating this new model in our already-existing cloth garment simulation framework [VOL 95] [VOL 97], in order to get the most out of the new creative potentialities that virtual simulation brings to anyone.

Acknowledgements

We are grateful the Swiss Fonds National pour la Recherche Scientifique for funding the projects related to this work, as well as to all the people who contributed to this work, for technical help and suggestions, illustration design and text reviewing.

Notation Index

i, j	Particle indexes.
c, d	Collision indexes.
A, B	Mesh elements involved in collision **c**.
M	Mass matrix of all particles of the system.
Mi	Mass of particle **i**, diagonal elements of the matrix **M**.
P, P', P"	Position, speed, acceleration vectors of all particles of the system.
Pi, Pi', Pi"	Position, speed, acceleration of particle **i**, elements of the vectors **P**, **P'**, **P"**.
Sci	Barycentric coordinates of the contact points of the collision **c** relative to the particle positions **Pi** (taken positive for the **A** element, and negative for the **B** element).
Qi	Geometrical property of particle **i** (being either position **Pi**, speed **P'i**, acceleration **P"i**).
Qc	Geometrical property of the collision **c**, weighted sum of the geometrical properties **Qi** of the particles involved in the collision (2).
Δ_0**Qc**	Expected geometrical property correction of collision **c**, computed from **Qc** using the collision correction model.
δ**Qc**	Practical correction of collision **c**, which, combined, should yield the expected correction for all collisions simultaneously.
δ**Qci**	Geometrical property correction of collision **c** on a particle **i**, obtained by distribution of δ**Qc** on all particles involved in the collision (4).
Δ**Qi**	Total geometrical property correction exerted on particle **i** by all collisions, being the sum of all δ**Qci** on that particle (12).
Δ**Qc**	Effective geometrical property correction of collision **c**, resulting from all Δ**Qi** contributions (9).
Mc	Reduced mass of collision **c** (7).

Bibliography

[BAR 92] : **D. Baraff, A. Witkin**, *"Dynamic Simulation of Non-Penetrating Flexible Bodies"*, Computer Graphics (SIGGRAPH'92 proceedings), Addison-Wesley, 26(2), pp 303-308, 1992.

[BAR 96] : **D. Baraff, A. Witkin**, *"Linear Time Dynamics Using Lagrange Multipliers"*, Computer Graphics (SIGGRAPH'96 proceedings), Addison-Wesley, pp 137-146, 1996.

[BAR 98] : **D. Baraff, A. Witkin**, *"Large Steps in Cloth Simulation"*, Computer Graphics (SIGGRAPH'98 proceedings), Addison-Wesley, 32, pp 106-117, 1998.

[BRZ 88] : **R. Barzel, A. Barr**, *"A Modeling System Based on Dynamic Constraints"*, Computer Graphics (SIGGRAPH'88 Proceedings), pp.179-187, 1988.

[CAM 97] : **S. Cameron**, *"Enhancing GJK: Computing Minimum and Penetration Distances Between Convex Polyhedron"*, IEEE International Conference on Robotics and Automation,1997.

[EBE 96] : **B. Eberhardt, A. Weber, W. Strasser**, *"A Fast, Flexible, Particle-System Model for Cloth Draping"*, Computer Graphics in Textiles and Apparel (IEEE Computer Graphics and Applications), pp 52-59, Sept. 1996.

[FUD 97] : **I. Fudos, C. Hoffmann**, *"A Graph-Constructive Approach for Solving Geometric Constraints"*, ACM Transactions on Graphics, pp 179-216, 1997.

[JOU 96] : **A. Joukhadar, A. Wabbi, C. Laugier**, *"Fast Contact Localisation Between Deformable Polyhedra in Motion"*, Computer Animation'96 Proceedings, pp 126-135, 1996.

[KLO 97] : **J.T. Klosowski, M. Held, J.S.B. Mitchell**, *"Efficient Collision Detection Using Bounding Volume Hierarchies of k-dops"*, IEEE transactions on Visualization and Computer Graphics,4(1), 1997.

[LAF 91] : **B. Lafleur, N. Magnenat-Thalmann, D. Thalmann**, *"Cloth Animation with Self-Collision Detection"*, IFIP conference on Modeling in Computer Graphics proceedings, Springer-Verlag, pp 179-197, 1991.

[PRE 92] : **W.H. Press, W.T. Vetterling, S.A. Teukolsky, B.P. Flannery**, "Numerical Recipes in C", Second edition, Cambridge University Press, 1992.

[SNY 93] : **J.M. Snyder, A.R. Woodbury, K. Fleisher, B. Currin, A.H. Barr**, *"Interval Methods for Multi-Point Collisions between Time-Dependant Curved Surfaces"*, Computer Graphics annual series, pp 321-334, 1993.

[VOL 94] : **P. Volino, N. Magnenat-Thalmann**, *"Efficient Self-Collision Detection on Smoothly Discretised Surface Animation Using Geometrical Shape Regularity"*, Computer Graphics Forum (Eurographics'94 proceedings), Blackwell Publishers, 13(3), pp 155-166, 1994.

[VOL 95] : **P. Volino, M. Courchesne, N. Magnenat-Thalmann**, *"Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects"*, Computer Graphics (SIGGRAPH'95 proceedings), Addison-Wesley, pp 137-144, 1995.

[VOL 97] : **P. Volino, N. Magnenat-Thalmann**, *"Developing Simulation Techniques for an Interactive Clothing System"*, Virtual Systems and Multimedia (VSMM'97 proceedings), Geneva, Switzerland, pp 109-118, 1997.

[VOL 00] : **P. Volino, N. Magnenat-Thalmann**, *"Implementing Fast Cloth Simulation with Collision Response"*, Computer Graphics International'2000 proceedings (to appear), July 2000.