Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2004)
R. Boulic, D. K. Pai (Editors)

Collision Between Deformable Objects Using Fast-Marching on Tetrahedral Models

D.Marchal[†], F. Aubert[‡], C. Chaillou[§]

Alcove-INRIA Futurs, LIFL (UMR CNRS 8022), Université Lille I.

Abstract

This paper presents an approach to handling collision between deformable objects using tetrahedral decomposition. The tetrahedral volumetric model is often used to simulate deformable objects that handle cuts and splits. Interaction between such objects in a complex environment is still an open problem in interactive simulation. This paper is mainly focused on obtaining a fast computation of a reliable penalty response. The method consists in using an approximated distance map to compute a penalty based response. We propose to compute the distances to the boundary using a modified "Closest Point" algorithm derived from Fast Marching. The presented algorithm, inspired by the [FL01], has the advantage of computing rapidly the "Closest Point" in the volumetric tetrahedral mesh without any use of an additional computation grid. From the resulting distance map a response is computed using a new "segment-in-object" response that offers more reliable results than the "point-in-object" generally used in previous works. Using this collision model, simulation at interactive rate can be considered in an environment composed of objects that can be deformed and cut.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation, I.3.5 [Computer Graphics]: Physically based modelling

1. Introduction

It is well known that real-time physically based animation needs efficient collision detection as well as an accurate penalty response to it. Complex environments, such as those needed for virtual surgical operations, require advanced physical models for many objects and robust integration methods to compute the motion interactively. The time dedicated to the collision detection and the collision response computation should be reduced as much as possible. Many efficient solutions exist for rigid bodies but collision between deformable objects is still a problem. All classical assumptions on the rigid shape (convexity, heavy precomputations) can not be made anymore, and the methods can not be adapted or require slow update processes. In addition to deformable objects, which only involve homeomorphic changes, cutting handling implies topological changes that make the collision problem more complex.

For such deformable objects, the context of this paper is focused on a penalty based response deduced from the assumption that 3D object overlaps at a given time which is known as a *static* approach or *3D intersection* method. This approach has the well-known drawback that collision between two thin parts of objects can be missed. Yet it can be tolerated in many situations by considering a coherence between the shapes and the time-steps. We choose this context because it is faster than a continuous method or than the contact time determination.

In the context of collision detection, most solutions are only based on primitives (spheres, triangles, tetrahedrons). These solutions are efficient but actually do not suffice for a reliable response (we call these pure primitive based methods "local" models). Actually, a reliable response requires information from the whole object (i.e. the "global model") such as distance fields and/or volume intersection [KLM02]. We propose a compromise method that takes advantage of primitives for the detection but without loss of knowledge

† damien.marchal@lifl.fr
‡ fabrice.aubert@lifl.fr
§ christophe.chaillou@lifl.fr

from the object for the response computation (we call this compromise a "semi-global" model).

This compromise can be obtained with a tetrahedral decomposition of objects. This kind of objects are often used in Finite Element Methods for instance, and can be considered for mass-spring systems. Many tools exist for their handling (the examples in this paper are obtained from the Gmesh program [GR]). In this paper, the detection phase consists in tetrahedron overlaps while the response is deduced from the computation of an approximation of the penetration depth. The required distance field must be obtained rapidly at each time step since the distance map changes during the deformation of the objects. The context of this work can be closely related to [FL01] in which a fast marching level-set method is used to compute, and update, the distance field of a deformable tetrahedral volume, and to [THM*03] in which the objects are represented with only one layer of tetrahedrons.

The main contribution of this paper is to propose a very fast approximation of the distance field calculation. The gain is obtained by removing the intermediate computation grid required for the fast marching method. Thus, the computation of the distance field can be made at each time step. The second important novel point of this paper is that the response is built from a segment-in-volume approach and not only from a point-in-volume approach (like in [FL01] or in [THM*03] for example). Considering only intruding points for the response is fast but it often leads to an incoherent response involving visible artifacts in animation. That is why tetrahedrons that overlap by edges have to be taken into account for the response. Furthermore, a *segment-based* response is a better convey of the entire volume intersection.

The paper is composed as follows: Section 2 presents the previous works in the context of penalty based methods. In Section 3 the whole method is overviewed including the collision detection process we adopt. Section 4 details the approximated fast-marching algorithm to compute the distance field it is followed in Section 5 by the computation of the response force. Section 6 illustrates the results with complex animations of deformable objects.

2. Background

Three cases of collision between deformable objects can be considered: object-vs-object, object-vs-tissue and tissue-vs-tissue. When objects have a thickness, a response to the collision can be computed from an intersection measure. The most used intersection measures are the intersecting volume and the minimal translation distance that separates two overlapping objects, also called Penetration Depth (PD).

Many collision models exist, and the general approaches are here classified in three catergories. The first one is based on a global view of the objects, the second one considers objects as a sum of primitive elements while the last one tries to combine the advantages of the two previous approaches using primitive elements and computing a global collision response. The presented work is in this third category.

2.1. Global collision methods

We classified in this first category the collision models that treat objects as entities and not only as a bunch of primitives. Efficient collision detection between geometric objects can be found in [LMP94] [Cam97]. They are based on topology and adjacency information to rapidly compute whether two objects intersect. As the computation is based on incremental computation, they can be combined with a temporal coherence system that initializes the new search from the previous frame results.

Once the collision is detected, a PD can be computed with [vdB99] or [KLM02] (for example). These algorithms only work on convex objects. To handle non-convex objects, an additional step is needed to break the object in convex pieces like in [EL01]. This step would be too expensive in a real-time context.

A little apart, methods based on the intersection volume can be found. The intersection volume is generally considered as a better but slower measure of penetration [OH99]. It is better because the entire interpenetration zone is involved and thus it can be considered more physically consistant. It is slower because volume measure is computed with "brute-force" approaches (explicit reconstruction or voxel based approach). These methods perform very poorly except for some recent graphics hardware accelerated solution as in [HTG03]. These discreet models handle the deformable objects in a straightforward manner and are well suited for GPU based implements.

2.2. Local collision methods

We classified in this second category collision models that treat objects as union of elementary primitives. The collision detection and the collision response are performed using a per primitive collision test without any need of "extra" information.

In [DMC02] spheres are used and the response is computed by summing the independant contributions of each couple of spheres in collision. The response does not depend on the sphere positions in the object, and overlaps of the spheres in the object can give incoherent responses. Updating a sphere approximation of a deformable volume object can induce costly computations.

In [GRLM03] the authors present a graphics hardware accelerated method to detect collision of objects composed of surface triangles. Building a response from the surface elements only is complicated.

A local model dedicated to tetrahedrons is found in [THM*03]. The collision detection and response are based

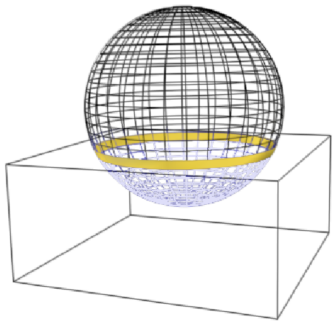

Figure 1: *In gray (yellow), the intersecting faces. In light gray (purple) the whole intersection that has to be detected to have a realistic deformation.*

on a *point-in-tetrahedron* test which means that no response is produced when objects intersect only by edges. To reduce the number of intersection tests the collision is restricted to only one layer of tetrahedrons (that gives a thick surface for the object boundaries).

Such local models seem to be suited for collision between deformable objects because no assumption about the shape is made. But computing a response from such a local model is generally less simple since a robust penetrating measure is difficult to construct.

2.3. Semi global methods

Finally we classified in this third category the collision models that combine advantages of both global and local methods. These methods generally mixe the robustness of a "real" penetration depth computation for response with a primitive overlap test that does not need topology assumption.

Figure 1 shows the difference between a local collision system and a semi-global one. The gray band is the result of a local collision. A response based on it does not produce realistic deformation behavior because the full Contact Area (in light gray) and all intersecting mechanical points are needed. This picture is from [SL00] where the full contact area is computed by categorizing for each face whether it is inside or outside of the other object. In the idea of categorizing which elements of the object are *inside* or *outside* the intersection area, [BWK03] introduces a collision system for tissues.

A semi-global approach based on tetrahedrons to handle the collision in a static envionment can be found in [Gei00]. For each intruding point, the PD is computed by finding the face that probably crosses during the last time-step. The localization of the face is done by navigating through the tetrahedral mesh following the line between current point position and the previous point position.

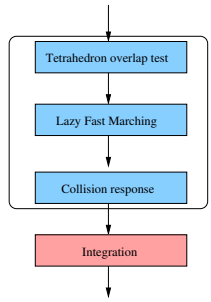

Figure 2: *Pipeline of collision*

The work in [FL01] is also a semi-global method. Each tetrahedron of an object stores the distance to the mesh boundary on its four nodes. These distances are computed with the Fast Marching method [Jam96] and are interpolated to approximate the whole depth map of the object. An intersection test is done locally between pairs of tetrahedrons and the approximated depth map is used to compute the response. We propose a faster approximation of this distance field.

3. Overview of the method

The collision model is divided in three steps (Figure 2).

The first one is collision detection. The collision detection uses an overlap test of the tetrahedrons like in [FG03]. Many choices can be made to accelerate the primitive-primitive test with a broad phase. With deformable objects, the detection acceleration is generally based on techniques as voxels grids, hash-table [THM*03] or sweep and prune [CLMP95]. These techniques can be also combined with some simple Bounding Volume Hierarchies (BVH) like AABB [vdB97]. The key point of those acceleration methods is that they have to be updated quickly enough during object deformations.

The broad phase is outside the topic of this paper and we choose a straightforward method based on a voxel grid. Each tetrahedron is rounded by a sphere and placed in the grid to fast reject the non-overlapping tetrahedrons.

The last two steps of the collision pipeline are detailed in the next two sections. A "Lazy Fast Marching" step is followed by the computation of the penalty based response that prevents further penetration. This penalty is computed with a classical *point-in-object* strategy or with a more precise *segment-in-object* strategy. An argument about point *vs* segment response in the context of rigid body stacking can be found in [GBF03].

4. The Fast Marching method

The Fast Marching is a fast algorithm introduced by [Jam96] to solve the Eikonal equation. Distance map computation is

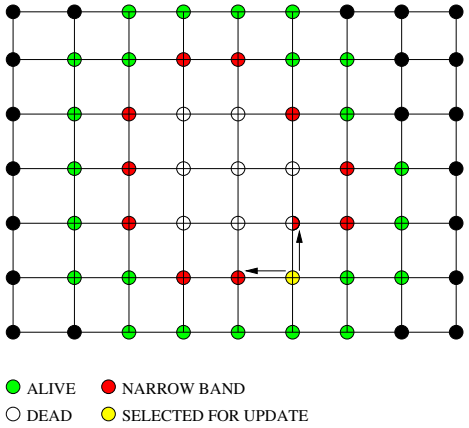

Figure 3: *A step of Fast Marching on an orthogonalized grid. A point is extracted from the heap and its neighbors are updated.*

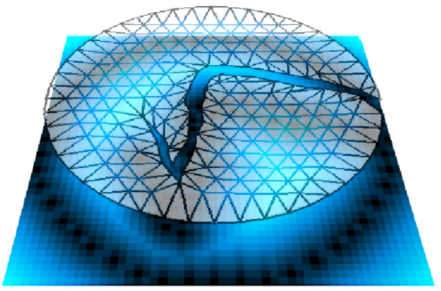

Figure 4: *An object (the disc) and its Fast Marching computation grid. It can be seen that some parts of the computation grid are useless as they are outside of the mesh object.*

a such Eikonal equation and an increasing number of papers seem to use it. We first describe the Fast Marching to compute an inner distance field like in [FL01] and then our new algorithm.

4.1. Fast Marching to compute a depth map

Let us describe the Fast Marching method as presented in [FL01] to compute the distance between each tetrahedron node of a volumetric object to the boundary of the object.

The native method is designed for an orthogonal grid (Figure. 3). Each point has a distance value d and a state (ALIVE, NARROW_BAND or DEAD). Initially the points that are part of the boundary are marked as ALIVE and their value is set to 0. Neighborhoods are marked as NARROW_BAND and their value is computed from the ALIVE points using a finite difference scheme (see [Jam96]).

After this initialization step the algorithm iteratively extracts the next point from a min-heap. This point has its status changed to ALIVE, meaning it has been fixed. For each not ALIVE neighbor of this extracted point, the status is changed to NARROW_BAND and the distance d is updated.

The update procedure uses a finite difference scheme designed to work on an orthogonal grid. To compute the distance on a volumetric mesh [FL01] proceeds in the following way:

- all points of a 3d grid are marked as DEAD;
- object surface is rasterized in this grid and forms the initial ALIVE points;
- the depth map is computed via Fast Marching on the 3D grid;
- the tetrahedrons' nodes read their d value from the grid with interpolation.

The grid resolution has to be set arbitrarily and is a trade off between speed and precision. In addition, the distance is computed for the entire grid without taking into consideration whether the point is inside or outside of the object, resulting in unnecessary computations. This can become a problem if the shape evolves a lot and the grid has to be adapted in consequence. For these reasons, the use of a standard Fast Marching to compute the depth map on volumetric mesh is not a fully satisfactory solution. We present a novel approach that directly uses the existing mesh structure to compute the distance.

4.2. Closest Feature Fast Marching

The key of Fast Marching efficiency resides in the usage of a Dikjstra's like graph traveling leading to a $O(n \log n)$ complexity with n the number of points. Two solutions to remove this grid and obtain faster computation of the depth map have been considered, keeping in mind this graph traveling approach.

The straightforward solution is to use the Fast Marching extension to unstructured triangulation presented in [RJ98] but the Update Function is much more complex and suffers from degeneracies. So we use a "Closest Point" principle based on Fast Marching. This new approach shares the same idea as [MB] or [Tsa02] where the Closest Point is propagated instead of the distance.

In our approach, which we called *Closest Feature Fast Marching*, each tetrahedron's node stores its closest boundary's feature (point, line, triangle). The boundary nodes have their distance value set to 0, are marked as ALIVE, and have their closest feature set to themselves. The algorithm iteratively selects the ALIVE point with the least distance and updates its neighbors using its f feature. The neighborhood of a point is given by the incident edges. The update function

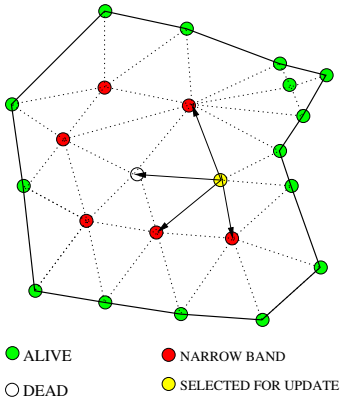

Figure 5: *The Closest Feature Fast Marching directly computes the distance using the mesh.*

is:

$$n_f = (n_d > dist(n_x, f))?f : n_f$$

$$n_d = (n_d > dist(n_x, f))?dist(n_x, f) : n_d$$

where n is a node at n_x position and at a n_d distance to the closest feature boundary n_f.

The algorithm may fail to report the closest feature and may select a farther one. The result is an overestimation of the distance but we do not find this problematic as the error does not propagate and the distance field still has the correct shape (i.e. it vanished on the object boundary) for our objective: computing a penalty response.

We have implemented two versions of the algorithm. The first one CFFM (for *Closest Feature Fast Marching*) saves the closest feature (point, edge, face) while the second one CPFM (for *Closest Point Fast Marching*) only uses the distance to the nodes of the boundary. The CPFM has a larger error but it improves the computation speed.

4.3. Results

The presented algorithm were implemented for the 2 and 3D cases. Computation time are shown in Figure 6. It can be seen that increasing the number of nodes has a linear influence on computation time. Other exemples of distance field computed on 3D mesh with up to 300 000 tetrahedrons are showed in Figure 7.

In addition we compare the CFFM class algorithm with our implements of a grid based Fast Marching and measure the computation time, the maximal error (maximal difference between exact value and computed value) and median error (sum of all error over the number of node). This test framework uses 2D triangle meshes dumped from a tissue simulation where objects are deformed and cut in various separated parts of non-convex shapes.

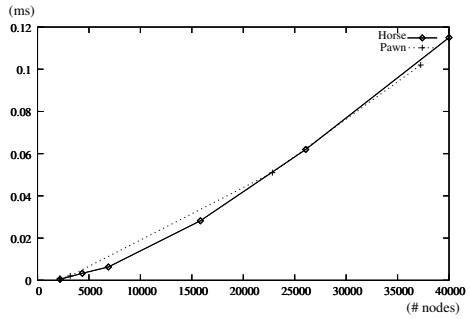

Figure 6: *Computation time (in ms) to approximate distance field with CPFM on object of increasing size.*

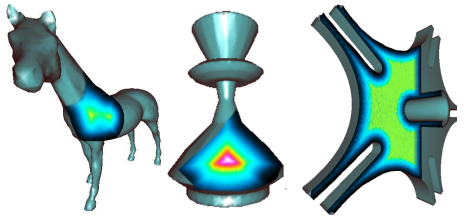

Figure 7: *The 3D distance field of large objects (horse, pawn, imprison) compose of 300000 tetrahedrons is computed with CPFM in 0.1s.*

Each test was made with mesh of different resolution from 200 to 4000 tetrahedrons node, they permit to conclude that:

- When the same number of tetrahedron nodes is equivalent to the number of grid cells, grid based approaches are faster than CPFM.
- Closest Point approaches have a much lower median error than numeric methods.

The advantage of the presented method is its hability to compute good distance approximation that directly depends on mesh resolution. This has a special interrest as low resolution meshes ($\approx$ 1000 nodes) as the ones used in realtime simulators. On such situtation the CFFM/CPFM algorithms are both faster than the other tested approaches and benefit from a reduced error.

It additionnaly makes the cutting unproblematic. If the mesh is cutted in multiple part, it makes no difference for the presented algorithm while a grid based approach would need a refitting strategy and a connectivity tracking to handle the differents separated parts.

4.4. Partial update of distance field

In previous examples, the distance field is computed for the whole object leading to a big number of unnecessary computation if used for collision response. We implement a partial

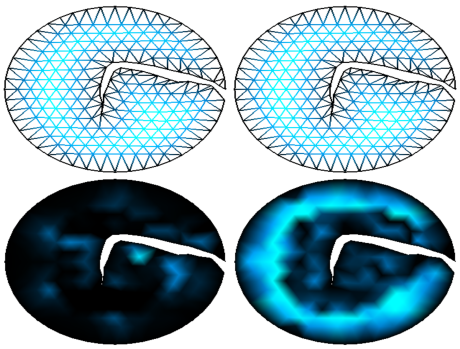

Figure 8: *A disk is deformed and cut. Left side is computed using the CPFM algorithm while the right side is computed using the Fast Marching on a 100x100 grid. The mesh and the computed distance field are drawn at the top with color scale. On the bottom line we represent mesh with the computation error, clearer values means more error.*

Algorithm	Time (*ms*)	max error	median error
EXACT	9.930		
FM 20x20	0.288	0.326	0.085
FM 100x100	9.728	0.149	0.011
CFFM	0.630	0.02	8.10^{-5}
CPFM	0.196	0.12	0.008

Figure 9: *Results of the differents algorithms for the disc object of 486 nodes.*

update of the distance field using the fact that collisions generally append in the vicinity of the object surface. This is done by marking the colliding tetrahedrons during detection and stop the CFFM loop when all theses colliding tetrahedrons have their distance computed. This decreases the computation time and makes the CPFM for collision a really inexpensive computation even on very big meshes. This makes the system fast in the common collision situation and robust to more rare deep collision.

5. The Collision Response

The detection, described in Section 3, returns pairs of overlapping tetrahedrons. From these overlapping tetrahedrons we implement two strategies to compute the penalty response forces using the previously computed depth-map.

The first strategy computes response force for the *Points in Objects* situation as in [HFL00] or [THM*03]. Only generating force on the intruding points leads to unpleasant artifacts as some colliding situations do not yield response force and thus penetration will not be stopped. To prevent those artifacts we define a response based on a *Segments in Object*

Algorithm	Time (*ms*)	max error	median error
EXACT	28.0		
FM 20x20	0.600	0.34	0.098
FM 100x100	12.0	0.081	0.016
CFFM	5.0	0.002	3.10^{-6}
CPFM	2.3	0.06	0.001

Figure 10: *Results of the differents algorithms for a rectangular object of 1000 nodes.*

force computation. In the following we present these two strategies.

5.1. Point in Object response

When a point p penetrates an object, the approximation of the penetrating distance d value is possible as a linear combination of tetrahedron nodes $x_{1..4}$ and distance values $d_{1..4}$. Let $u = u_{1..3}$ be the barycentric coordinates of p. These barycentric coordinates define the position of the point p in a frame whose origin coincides with x_4 and whose axis coincides with the edges of t adjacent to x_4. The coordinates in this new frame are computed as:

$$u = G^{-1}[p - x_4]$$

with the following 3 by 3 matrix:

$$G = [p - x_1, p - x_2, p - x_3]$$

To be in a tetrahedron a point has to fill up the following equations: $u_1 \geq 0, u_2 \geq 0, u_3 \geq 0$ and $u_1 + u_2 + u_3 \leq 1$

Linear interpolation of $d_{1..4}$ values is made in the following way:

$$\tilde{d} = u_1 d_1 + u_2 d_2 + u_3 d_3 + (1 - u_1 - u_2 - u_3) d4$$

To compute the force direction we use the gradient of the distance field. This gives an intuitive direction for the point to exit the object. In addition, as a linear interpolation is used inside the tetrahedron, the resulting gradient only depends on the nodes of the tetrahedron and not on the intruding point p.

This direction is computed from the $\tilde{d}$ formula as:

$$\nabla d = \begin{pmatrix} \frac{\partial \tilde{d}}{\partial x} \\ \frac{\partial \tilde{d}}{\partial y} \\ \frac{\partial \tilde{d}}{\partial z} \end{pmatrix} = G^{-1} \begin{pmatrix} u_1 - u_4 \\ u_2 - u_4 \\ u_3 - u_4 \end{pmatrix}$$

With these formulas we can compute, for each node penetrating an object composed of tetrahedrons, the approximate distance to the boundary. From this distance and its gradient a penalty is applied to the intruding point.

This point-in-object computation may miss some collision cases as on Figure 11 where the tetrahedrons collide by edges. This leads to unpleasant artifacts on low resolution mesh that leads us to introduce the segment-in-object approach.

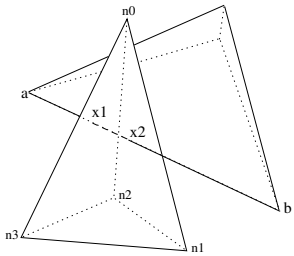

Figure 11: *Two tetrahedrons overlap, the segment [ab] intersects the other tetrahedron on points x_1 and x_2. In such situation a point based approach would not generate any response.*

5.2. Segment in Object response

The response force received by a segment intersecting a tetrahedron is computed with the following method.

For a line l intersecting a tetrahedron t, the intersection is defined by two line parameters λ_1 and λ_2. As the previously defined *Point in Object* response is linear for a given tetrahedron, the *Segment in object* response can be computed on the two intersection points and then integrated along the segment to take into account its length. Given a segment in parametric form:

$$x = a + \lambda(b-a)$$

the two end points are projected in the tetrahedron frame:

$$A = G^{-1}(a-t_4) \text{ and } B = G^{-1}(b-t_4)$$

this gives the following segment equation in the tetrahedron frame:

$$u = A + \lambda(B-A)$$

an intersection between this line and the tetrahedron is equivalent to test if for some λ included in $[0,1]$ the corresponding u fills up the following equations: $u_1 \geq 0, u_2 \geq 0, u_3 \geq 0, u_1+u_2+u_3 \leq 1$.

These tests give us the two points (x_1, x_2) corresponding to the parameters (λ_1, λ_2) that delimit the intersection. It is then easy to approximate the distances ($\dot{d_1}$,$\dot{d_2}$) using the *Point in Tetrahedron* formula.

Finally the distance is integrated over the segment. This integration is direct as it is equivalent to computing the area of a trapeze.

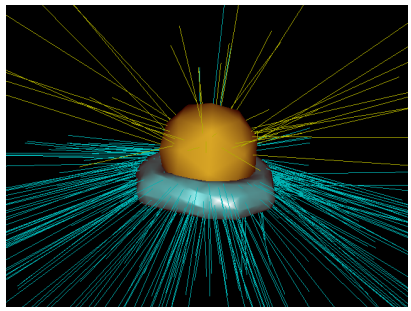

Figure 12: *Multiple collision points: a torus collides with a sphere. The computed responses forces are shown.*

$$\dot{d} = \int_{l \cap t} d = \frac{1}{2}(\dot{d_1} + \dot{d_2})||x_1 x_2||$$

On the intersection points (x_1,x_2) two penalty forces ($\overrightarrow{f_1}$,$\overrightarrow{f_2}$) are computed from this $\dot{d}$ penetration measure. These penalty forces have to be transfered to the objects mechanical points. The force transfer is a little bit tricky as we use the parameters λ_1 and λ_2 to weigh the penalty force received on each mechanical point. In our example the segment extremities a and b would receives the following force:

$$a_f = \lambda_1 * \overrightarrow{f_1} + \lambda_2 * \overrightarrow{f_2}$$

$$b_f = (1-\lambda_1) * \overrightarrow{f_1} + (1-\lambda_2) * \overrightarrow{f_2}$$

6. Results

The collision model is implemented in an existing simulation framework that includes a deformable objects model with a tetrahedron based mass-spring system. Collision between tetrahedral objects is handled with our collision system while collision between a tetrahedral object and a "triangle" based one is handled by using the *point-in-tetrahedron* and the *segment-in-tetrahedron* response when triangles intersect tetrahedrons (Figure 14).

6.1. Performance analysis

We compared the *point-in-object* and *segment-in-object* approaches. In our context the segment based response gives a better response with less sensible artifacts and a reasonable computation time increases (Figure 16). The *segment-in-object* is, as expected, slightly more time consuming (Figure 13) but it has also the interesting property to reflects penetration depth as well as size of contact area. In our simulation we select on or the other response with respect to the mesh density.

The lazy CPFM also greatly improve performances and

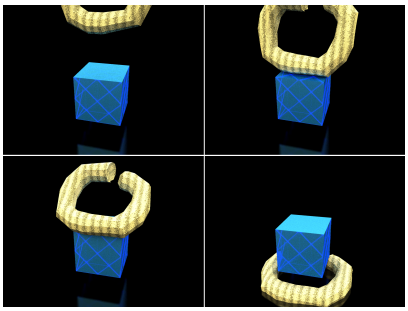

Figure 13: *A torus (7000 tetrahedrons) breaks on a cube. This scene was simulated in 2 minutes, the all collision process took 28 seconds with* point in object *response and 36 seconds when we use the* segment in object *response.*

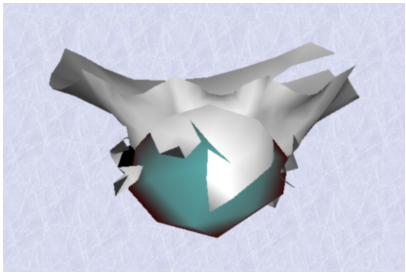

Figure 14: *Interaction between a non-tetrahedral object and a tetrahedral one. A deformable sphere falls on a tissue that breaks under the weight.*

makes the CPFM a really cheap computation compared to the rest of the collision pipe-line.

We also measured the collision time with respect to the number of colliding tetrahedrons and found it was linearly dependent.

6.2. Additional note

We noticed that tetrahedron with their four nodes on the boundary are problematic for both simulation and collision response. This problem was addressed in [MBTF03] where a meshing strategy has been purposed. In our simulator, those tetrahedrons are detected and handled by a simple subdivision. This increases the number of tetrahedrons and can become a problem when large breaks occur, generating a lot of invalid tetrahedrons (like explosions).

7. Conclusion

In this paper we have presented an interactive collision model for volumetric tetrahedral objects. Its novative points are a new *Closest Feature Fast Marching* algorithm to compute distance on a volumetric mesh, and a *segment-in-object* based response that reduces the visual artifacts.

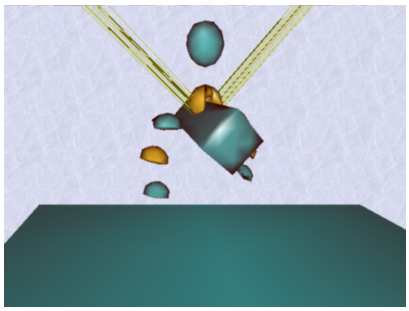

Figure 15: *Complex environment: elispoïds composed of 100 tetrahedrons are falling on a cube. They break on the cube's edge.*

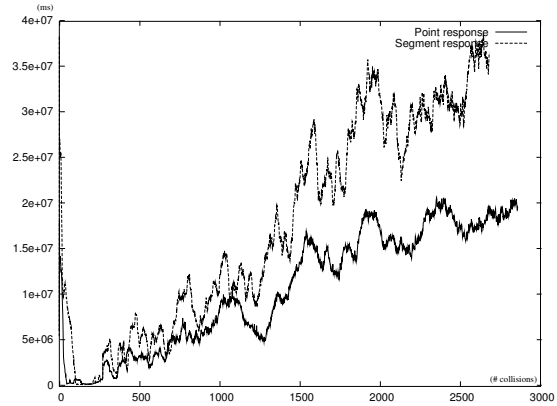

Figure 16: *Comparison between the total time spend (in ms) with respect to the number of colliding points. Highest curve stands for segment response.*

Comparing the presented *Closest Feature Fast Marching* with previous approaches shows it performs quite well in speed as in precision while the response based on the intersecting segment increases simulation realism. These results show that tetrahedron based collision systems are efficient enough to be used in real-time simulation. To continue this work, better collision detection and response has to be investigated as a lot of response computation are redudency. Voxel grid can be really inefficient for detection while the neighborhood relationship between tetrahedrons could be used to accelerate it. Remove those redundency would make the collision and the response a much more cheap operation.

8. Acknowledgements

This work has been carried out within the framework of the INRIA Alcove project and is supported by the IRCICA (Institut de Recherche sur les Composants logiciels et matériels pour l'Information et la Communication Avancée).

References

[BWK03] BARAFF D., WITKIN A., KASS M.: Untangling cloth. *Proceedings of ACM SIGGRAPH* (2003).

[Cam97] CAMERON S.: Enhancing gjk: computing minimum and penetration distances between convex polyhedra, 1997.

[CLMP95] COHEN J., LIN M., MANOCHA D., PONAMGI M.: I-collide: An interactive and exact collision detection system for large-scale environments. In *Proc. ACM Interactive 3D Graphics Conf* (1995), pp. 189–196.

[DMC02] DAVANNE J., MESEURE P., CHAILLOU C.: Stable haptic interaction in a dynamic virtual environment. In *Proceeding of IROS* (2002).

[EL01] EHMANN S. A., LIN M. C.: Accurate and fast proximity queries between polyhedra using surface decomposition. In *Proc. of Eurographics* (2001).

[FG03] F. GANOVELLI F. PONCHIO C. R.: Fast tetrahedron-tetrahedron overlap algorithm. *ACM Journal of Graphics Tools 7-2* (2003).

[FL01] FISHER S., LIN M.: Fast penetration depth estimation for elastic bodies using deformed distance fields. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2001).

[GBF03] GUENDELMAN E., BRIDSON R., FEDKIW R.: Nonconvex rigid bodies with stacking. In *SIGGRAPH* (2003), pp. 871–878.

[Gei00] GEIGER B.: Real-time collision detection and response for complex environments. In *Proc. CGI* (2000).

[GR] GEUZAINE C., REMACLE J.-F.: Gmesh. http://www.geuze.org/gmesh.

[GRLM03] GOVINDARAJU N. K., REDON S., LIN M. C., MANOCHA D.: Cullide: Interactive collision detection between complex models in large environments using graphics hardware. In *ACM SIGGRAPH/Eurographics Graphics Hardware Proceedings* (2003).

[HFL00] HIROTA G., FISHER S., LIN M.: *Simulation of Non-penetrating Elastic Bodies Using Distance Fields*. Tech. Rep. TR00-018, University Of North Carolina, 23 2000.

[HTG03] HEIDELBERGER B., TESCHNER M., GROSS M.: Real-time volumetric intersections of deforming objects,. *VMV Proceeding* (2003), 461–468.

[Jam96] JAMES A S.: A fast marching level set method for monotonically advancing fronts. In *Proceedings of the National Academy of Sciences* (1996), vol. 93,4, pp. 1591–1595.

[KLM02] KIM Y. J., LIN M. C., MANOCHA D.: Deep: Dual-space expansion for estimating penetration depth between convex polytopes. In *IEEE International Conference on Robotics and Automation* (2002).

[LMP94] LIN M., MANOCHA D., PONAMGI M.: Fast algorithms for penetration and contact determination between non-convex polyhedral models, 1994.

[MB] MAUCH S., BREEN D.: A fast marching method of computing closest points. http://www.cco.caltech.edu/ sean/closestpoint/closept.html.

[MBTF03] MOLINO N., BRIDSON R., TERAN J., FEDKIW R.: A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *12th Int. Meshing Roundtable* (2003), pp. 103–114.

[OH99] O'BRIEN J. F., HODGINS J. K.: Graphical modeling and animation of brittle fracture. In *The proceedings of ACM SIGGRAPH* (1999), pp. 137–146.

[RJ98] RON K., JAMES A S.: Fast marching methods on triangulated domains. In *Proceedings of the National Academy of Sciences* (1998), vol. 95.

[SL00] SUNDARAJ K., LAUGIER C.: Fast contact localisation of moving deformable polyhedras. In *IEEE Int. Conf. on Automation, Robotics, Control and Vision* (2000).

[THM*03] TESCHNER M., HEIDELBERGER B., MUELLER M., POMERANETS D., GROSS M.: Optimized spatial hashing for collision detection of deformable objects. *VMV Proceeding* (2003).

[Tsa02] TSAI Y. R.: Rapid and accurate computation of the distance function using grids. *Journal of Computational Physics 178*, 1 (may 2002), 175–195.

[vdB97] VAN DEN BERGEN G.: Efficient collision detection of complex deformable models using aabb trees. *Journal of Graphics Tools* (1997).

[vdB99] VAN DEN BERGEN G.: A fast and robust GJK implementation for collision detection of convex objects. *Journal of Graphics Tools: JGT 4*, 2 (1999), 7–25.