Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2010)
J. Bender, K. Erleben, and M. Teschner (Editors)

Edge collision detection in complex deformable environments

T. Jund and D. Cazier and J.-F. Dufourd †

LSIIT, UMR CNRS 7005
University of Strasbourg
France

Abstract

We present in this paper a simulation framework that allows a precise and efficient handling of collisions and contacts between deformable moving bodies and their environment. The moving bodies are sampled as meshes whose vertices are followed in a convex subdivision of the surrounding space. Particles are continuously spanned along the edges to detect collisions with cells of this subdivision.
Our method supports dynamic subdivision of the moving bodies and contact areas. It allows us to correctly handle geometric and topological changes in the environment, like cuts, tears or breaks and, more generally, additions or removals of material. We report experimental results obtained with mass spring and shape matching based physical simulations and discuss the performance of our method. We compare our approach with classical ones based on hierarchical data structures.

1. Introduction

Real-time collision detection for deformable moving bodies is still a difficult task for animations and physical simulations. This is especially true when the objects and their environment have to support topological changes. For instance, in medical simulations – especially in minimally invasive surgery – the moving bodies are scalpels, catheters or endoscopes and the environment is a virtual patient represented by models of organs, tissues or vascular networks. These tools are used to cut, tear or burn the tissues and may possibly cause changes in the topology of the environment. This includes removal of material, pieces of tissue coming off the environment and apparition of holes or tunnels.

In our framework, the environment defines a partition of the 3D space, modeled as a volumetric mesh. It includes objects of the scene the user may interact with, and the surrounding space. This mesh describes a set of convex polyhedrons tied by pairs, some faces of which are marked as impassable – the external boundary of the environment and the internal obstructions or objects. The moving bodies are displaced in this environment and cannot get over vertices, edges or faces of the impassable areas. The moving bodies are solids whose boundaries are sampled as vertices connected by edges. They are modeled as manifold meshes. Both environment and moving bodies are deformable and support topological changes. Those changes can be caused by the interaction (tears, cuts and breaks) or by the mechanical simulation if adaptive refinement of the meshes is needed.

Classical collision detection algorithms provide estimations of the interpenetrations volumes that are usually translated in the mechanical subsystem as penalties forces. This collision detection scheme is not sufficient in these kind of simulations where the contacts between the tool and the environment should be precisely acquired to provide the best visual and haptic feedback.

The collision detection (CD) system we present here ensures that, at any time, the vertices or edges of the moving bodies do not get over the environment faces and thus do not enter the impassable areas. Our CD system exactly finds all collisions occurring between the edges of the moving bodies and the environment. It provides a robust computation of the time and location of the collisions, as well as their local geometric configurations (contact zones and normals, impact points, friction conditions, etc.). This information is

† Supported in part by french ANR as VORTISS project.

DOI: 10.2312/PE/vriphys/vriphys10/069-078

made available to the physical subsystem that in turn computes the responses in terms of reaction forces and velocity.

However, the faces of the bodies are not considered for the collision detection – i.e. intersections between faces of the bodies and the environment are not searched. This is the compromise we choose to obtain real-time responses with accurate edge/edge collision detection. This lack of precision is easily bounded with appropriate increases in the sampling resolutions of the moving bodies. Note that in many applications – and especially medical ones – the environment is smooth enough for this restriction to be acceptable. Actually, this corresponds to small penetrations of vertices of the environment into the faces of the moving bodies. However the inverse, penetration of the moving object in the environment, is not allowed in our system.

Within this context, usual methods for collision detection – based on spatial subdivisions of the environment – are pushed to their limits. The used data structures – usually trees – have to be updated frequently which is not compatible with real-time applications dealing with large environments. The main contribution of this paper is a collision detection framework supporting changes in the topology of both moving bodies and environment in real-time.

In the following, section 2 presents related works and discusses their advantages and limitations in this context. Section 3 gives an overview of the context of the framework. Section 4 describes edges collision detection. Section 5 demonstrates how deformations and topological changes are handled. Section 6 gives edge tests elimination techniques. Section 7 presents our implementation and results obtained on a benchmark simulation. Section 8 gives a conclusion and tracks for future works.

2. Related works

Many algorithms and methods have been proposed to report collisions between moving objects. We refer the reader to existing states of the art [LM04, TKZ*04]. As self-collision is not the core of the current article, we refer the reader to [GKJ*05,CTM08,TCYM09] for an overview of some recent optimizations in this domain. We present and discuss here most representative collision detection approaches.

Bounding volumes hierarchies or BVH – are widely used to cull non-overlapping primitives. Well known results exist with BVH made of AABB [CLMP95], OBB [GLM96], k-dop [KHM*98] or spheres, depending on the aimed at applications. However, when deformations occur, additional computations are needed to update those tree structures [vdB97] which is usually a bottleneck for real-time simulations. Furthermore, the use of BVH for continuous collision detection (CCD) can lead to a very high number of false positive tests. Some works [TMT10] focus on the elimination of these false positives.

Other methods, motivated by time-critical collision detection constraints [Hub95], consider multiresolution contact handlings [OL03]. Stochastic [GD04] and evolutionary [JCA06] approaches have been proposed aiming at the same goal. All those methods are very efficient at handling large deformable scenes, but they lack precise contact information and may miss some collisions. That is not acceptable in our medical context where all contacts between tools and organs should be reported.

Image-space approaches [JH08] have the advantage of effectively handling deformable objects and dynamic environments without the need of pre-processing. The limitation of these kind of methods is the image-space resolution needed for its effectiveness. Here again, that makes them not suitable for applications requiring accurate collisions reporting.

More specialized works focus on endoscopies or angioscopies simulations [Gei00, LCDN06]. In these approaches the moving bodies consist in discrete sets of points or segments depending on the accuracy required by the simulation. The method presented in [LCDN06] is based on a tree of cells modeling sections of the vascular network. This approach is limited to tree-like structures and does not support topological changes (like cutting, ligaturing or tying). The method also needs a second level of details to model the skin of the vessels.

Work in [Gei00] is based on a tetrahedral decomposition of the mesh and uses temporal coherence. This algorithm supports non-tubular structures, but imposes restrictions on the topological changes supported within interactive time because of the costly updates of tetrahedrons and the lack of efficient neighbourhood information. In [THM*03] a hash function is used to map the tetrahedral subdivision and handle deformable objects. Another drawback of these methods is that the used tetrahedrons explicitly exist, leading to high memory costs not suitable for large and complex scenes.

Our method is similar to those discussed in [Gei00, LCDN06], but is more general and applies to deformable environments made of convex polyhedrons containing inner structures. It supports topological changes in both moving bodies and environment.

Finally, our method uses a particle-based collision detection where the vertices of the moving objects are followed up in the environment partition. Tetra-trees [JFSO06] is an example of such system handling any polyhedral decomposition at the cost of managing overlapping tetrahedrons. We extend the method presented in [JCD09] based on a convex decomposition as it better suits our models.

3. Method overview

3.1. Framework organization

The simulation framework works as follows. The kinetic data – position, velocity and contact information – is attached to the vertices of the moving bodies. Mechanical data

is attached to the cells of all meshes (environment and bodies). For instance, we associate stiffness to edges for mass-spring simulations and rest positions to vertices for shape matching based simulations. More evolved mechanical models may be used but this is out of the scope of this paper and is not further detailed.

In this framework, contact information is defined by its position and configuration (plane, edge or vertex) with the corresponding surface normal. The contact history can be used to determine if the object is sliding, bouncing or entering in a lasting contact.

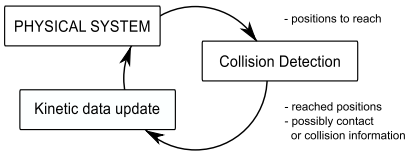

Figure 1: *The simulation framework.*

During the simulation loop (figure 1), the physical subsystem computes the involved forces from the positions, attached data and the contact information – for the reaction forces – and returns the positions the vertices should reach. Then the collision detection framework tries to move each vertex in turn and reports the reached positions and possibly collision or contact information. The position and velocity are modified according to the collision state and given back to the physical subsystem for the next loop.

3.2. Spatial decomposition

The environment, the bodies are moving within, is modeled as a partition of the 3D space in distinct cells: vertices, edges, faces and volumes. To simplify and optimize point/cell inclusion tests, a convexity constraint is required for every cell of this partition. The volumes are assumed to be convex polyhedrons and their faces convex polygons. The cells that lie on the exterior boundary or that represent inner obstacles are marked with a special *impassable* flag.

It is common usage to model such scene with simplexes, i.e. triangles and tetrahedrons. That leads to very large set of simplexes to represent realistic scenes, and thus too high memory costs. Our method handles any convex polyhedral decomposition and thus requires less memory for the volumetric representation. Polyhedrons whose shapes best fit the object geometry (like hexahedrons, prisms, pyramids, and so on) can be used as needed (figure 2).

In order to efficiently follow the vertices and to be able to handle deformations and subdivisions of its underlying mesh, the environment is structured with strong neighborhood information. We use the well known combinatorial map topological model [Lie94]. Its optimality and efficiency has already been demonstrated in geometric modeling [Duf89] and computational geometry [CD99]. Similar

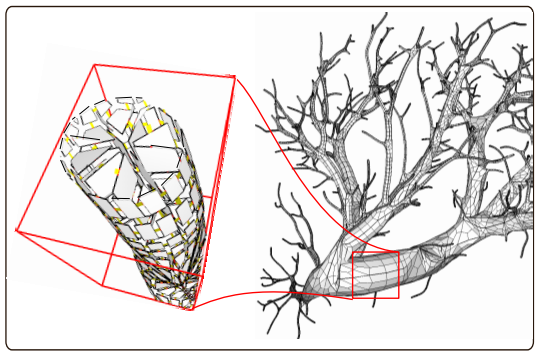

Figure 2: *A vascular network subdivided in various convex polyhedrons (about 15k polygons for 30MB in memory).*

topological models have been used, in the domain of physical simulation, to treat specific operations such as gluing and cutting [BGTG03, LT07].

3.3. Following particles

Our collision detection framework is based on particles moving in the environment. When a vertex is moved, a particle is thrown from its current position to its destination and, from there, particles are spanned along adjacent edges to neighboring vertices. This principle will be detailed further, but requires efficient collision detection between a particles flow and the environment.

The requirements are the following. To benefit from spatial and temporal coherence, each particle must maintain a state $S_t = (p_t, k_t, i_t)$ where p_t is its position, k_t is the kind of the cell it lies in (volume, face, edge or vertex), and i_t is an aimed at cell – i.e. the first cell in the environment intersecting its (linear) trajectory.

The displacements of a particle are followed up in the spatial decomposition of the scene and its state is updated as polyhedrons, faces or edges are traversed during its move. When a collision occurs, the returned state report the collision point p_t and the cell it is colliding with i_t. From this we can compute the contact information and normal.

We extend here the particle-based collision detection (PCD) forecast mechanism from [JCD09]. Any other PCD, such as [JFSO06], can be used. The extension to edge collision detection presented hereafter only requires the PCD algorithm to report the above mentioned data.

We propose in the following some improvements for multiple objects collision detection and an optimization to limit the edge CD overcost. This improvements require the PCD algorithm to be able to report all cells encountered during a displacement.

4. Edge collision detection

Particle-based collision detection is a flexible approach that can effectively handle large data in meshless simulation. By nature, it misses collisions between the surface of the moving bodies and the environment and especially along sharp features like edges. Some methods propose more dense and larger sampling entirely covering objects with particles [BYM05], but does not completely resolve this issue.

We present an extension of particle collision detection to edge/environment collision detection. A particle is associated to each vertex of the moving bodies and additional temporary particles are created as need to check for edges collision. We first present the 2D case that best supports plane figures, before detailing the general 3D case.

4.1. Sweeping edges: the 2D case

Consider an edge between two vertices associated with particles p_t and q_t. When p_t moves towards p_{t+1}, this edge sweeps over a small triangle corresponding to an elementary displacement. A collision occurs if this elementary triangle τ intersects an impassable cell (vertex, edge, face of the partition). As the environment is modeled by a continuous space partition, this happens if and only if one of the three edges of τ intersects an obstacle or if the obstacle is completely included in τ.

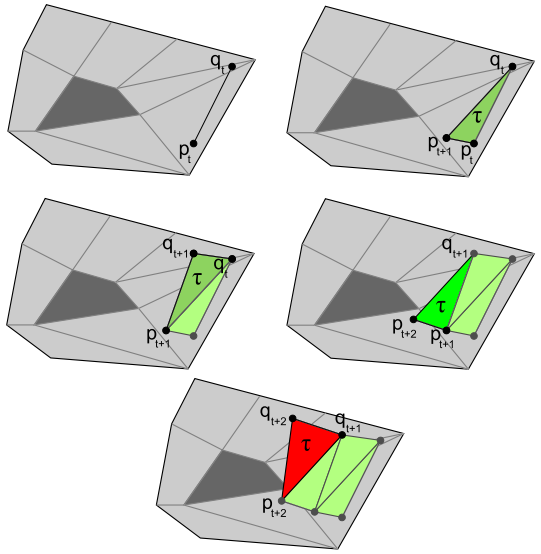

Figure 3: *Successive edges displacements and the elementary sweeping used to detect collisions.*

In the following we assume that no obstacle is smaller than the largest elementary triangular displacement. This size is proportional to the displacement length (distance from p_t to p_{t+1}) and to the sampling interval (the distance between two particles). In usual physical simulation this condition naturally holds and should not limit the domain of application of the method. The complete edge displacements are handled by alternative moves of the two particles at its extremities.

The three edges of τ are to be checked. The first one, $[p_t, q_t]$ is free of collision thanks to temporal coherence (a possible contact has previously been handled). The second one $[p_t, p_{t+1}]$ is checked by moving the particle associated to the vertex from p_t towards p_{t+1} using one PCD iteration. The third edge $[p_{t+1}, q_t]$ is checked by spanning a temporary particle from p_{t+1} to q_t with a second PCD iteration. Then q_t moves toward q_{t+1} with a first PCD iteration and a second one from q_{t+1} to p_{t+1} (figure 3). When a collision occurs, the vertex order influences the collision response at a magnitude depending on the chosen timestep.

4.2. Obtaining edge/edge contact information

When a collision occurs additional work is needed to precisely report the contact information. We first detail how the contact information can be retrieved, then we detail the possible responses – already mentioned in section 3 – whose details depend on the physical model used.

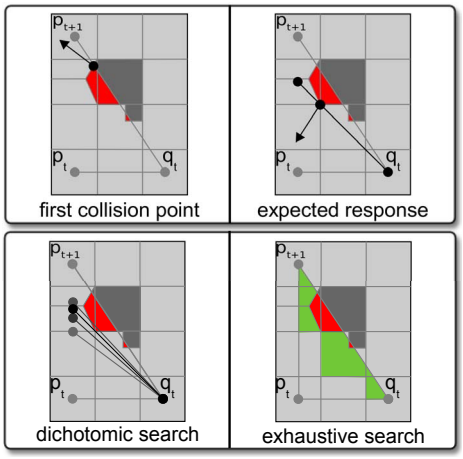

Figure 4: *Retrieving contact position and normal*

Whenever an edge collision is reported, the PCD iteration returns the first impassable cell encountered along the edge. This indicates a collision with an obstacle but does not give enough information about the impact point (figure 4). If precise contact information is needed, two methods can be used to retrieve it.

The first one is a dichotomic search. It iterates along the $[p_t, p_{t+1}]$ edge until the nearest segment with no contact is found. With bounded depth iteration, this method has low overhead cost. It reports approximated, but numerically reliable, contact information with accurate time of collision.

The normal is computed at the vertex of the last intersected dart (figure 4 bottom left).

The second method is an exhaustive search that should be performed when exact contact information is needed. It consists in searching all cells that are traversed by the PCD iteration for obstacles and to list their vertices. We use a special PCD iteration that does not terminate with the first encountered collision and continue until q_t is reached. The vertex with the smallest angle is reported as the contact point and used to compute the normal. The search can be limited to the vertices that lie below segment $[p_{t+1}, q_t]$ (in green figure 4 bottom right).

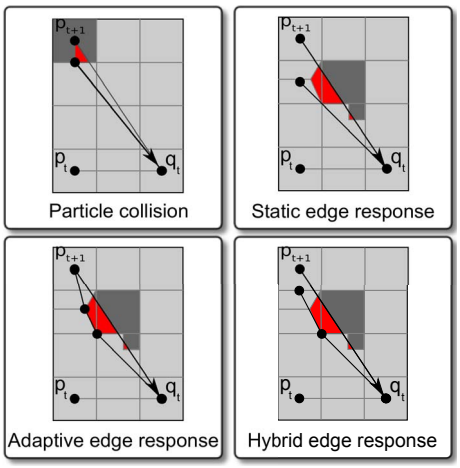

Figure 5: *Different kinds of responses to collisions*

Different responses are possible when collisions occur. The chosen one depends on the desired physical properties and on the ability of the mechanical model to support adaptive mesh subdivision. We detail here some possible responses depicted in figure 5.

When the collision simply involves a moving vertex (top left), its movement is stopped at the contact point. When an edge collision occurs, static or adaptive responses are possible. The static one also consists in stopping the edge movement at the contact point (top right).

The adaptive responses consist in inserting new particles at the contact points that implies folding the edge. These responses are adapted to deformable bodies with adaptive mechanics. If dichotomic search is used to retrieve contact information, a unique vertex is added. If an exhaustive search has been used, either the first contact vertex is inserted alone or, if several vertices are found, these are inserted simultaneously (bottom left).

When only one vertex is inserted, the two new subdivided edges are checked for collision because more than one collision can happen during one displacement. For this second PCD pass the static response – leading to a hybrid response (bottom right) – or the adaptive one can be chosen depending on the simulation needs.

The PCD naturally supports dynamic addition of particles. The only required updates concern the edges between particles which is a local operation. Attention should be paid not adding too many particles. In our experimentations we use the adaptive responses only if the lengths of the subdivided edges are longer than 25% of the colliding edge. More subtle heuristic can be used depending on the application.

4.3. Sweeping meshes: the 3D case

The previously described principle directly applies to the 3D simulations. The vertices of the mesh are moved successively and temporary particles are spanned along the edges as shown on figure 6.

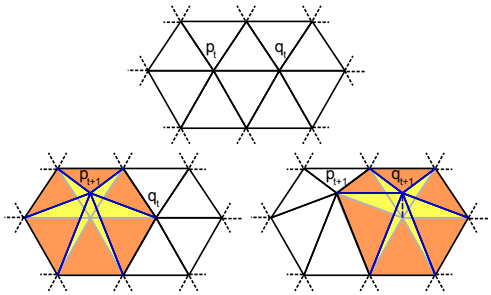

Figure 6: *Successive edges displacement for a 3D mesh.*

The collision between the edges of the moving bodies and the environment are precisely reported while the environment contains no relatively small obstacles. The same methods are used to retrieve the contact information. The resulting positions of the set of particles is then transmitted to the physical model.

The main difference lies in the ability to report all collisions. In the plane all collisions are detected as soon as no small obstacles are present. In dimension 3, things are more complex and only edges/environment collisions are reported. Figure 7 sum up the different geometrical configurations of collisions.

When the obstacles present sharp features (schematized by blue polyhedrons in figure 7), the corresponding vertices may interpenetrate the moving bodies through faces.

The largest penetration possible corresponds to an inversed pyramid resting on the moving mesh edges. The volume of this pyramid is bounded by the face of the moving bodies and by the displacement length of a single vertex. The solid angle of this vertex also influences this interpenetration volume. The penetration can be more important when the obstacle forms a sharp spike.

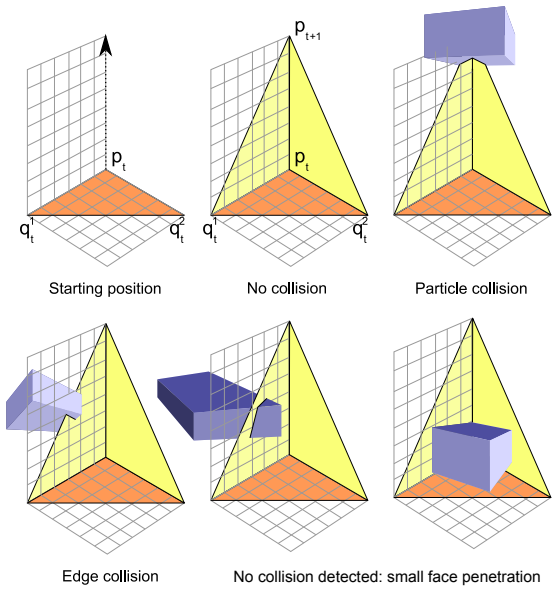

Figure 7: *Edges collisions cases and missed faces collisions*

In applications when the environment is made of soft material, with few sharp features, and when a sufficient sampling of moving bodies exists, the maximum interpenetration volume can be reasonably bounded. Moreover, in many applications the moving bodies are precise tools or object with fine sampling, whereas the environment is more roughly meshed. This makes the interpenetration possibilities less sensitive.

4.4. Multiple objects and self-collision

The cell subdivision allows to test for collision detection between objects or even for self-collision. For this purpose, each edge is registered within its containing cells. This way, the cells naturally act as bounding volumes, that by construction follow the environment deformations. Primitives included in the same cell are pairwise tested, with the usual intersection tests.

5. Deforming the environment

When the moving bodies are in a deformable environment, the simulation proceeds in two steps: first the vertices of the moving bodies are moved; then the vertices of the environment are moved while the bodies are kept in fixed positions. The transformations of the 3D mesh that models the environment are twofold: geometrical transformations preserving the cells topology and convexity; and topological transformations including remeshing to restore the convexity.

5.1. Geometrical deformations

Recall that every particle maintains a state $S_t = (p_t, k_t, i_t)$ used as forecast data to speedup the search of colliding cells during the next move. Whenever the environment is deformed we only have to check the validity of the stored forecast data and to update it if needed. Tree cases arise:

- **General case:** p_t still lies in the cell of i_t and aim at it. Nothing is done.
- **Changing orientation:** p_t still lies in the cell of i_t, but do not aim at it anymore. Nothing is done as the next PCD iteration will directly correct this.
- **Changing cell:** p_t no more lies in the cell of i_t (a face or an edge of the environment passes over p_t). The correct data is recovered by casting a particle from the center of the cell of i_t (the previously correct cell) to position p_t.

In practice testing if the third case applies requires a point/face orientation test for each face of the current cell. Thus the third case is always used. A PCD iteration with the state $S_t = (c, k_t, i_t)$, c being the position of the center of the cell i_t, toward p_t correct the forecast data with only 4 orientation tests if no changes is needed.

At this point the particles forecast data is correct. The last point to check is that the edges of the moving bodies are not in collision. That is achieved with the same spanning strategy as before. A particle is casted along every edge. Whenever an edge collision is reported the same kind of responses can be used. For rigid object the vertices of the colliding edges are moved in a correct position. For deformable dynamic object, particles can be inserted subdividing the faulty edges.

5.2. Topological deformations

The model we use for the environment allows local changes in the topology of the partition. We describe here some basic topological operations and detail how to update the particles states to correctly manage these transformations.

Sewing: When the cell containing a particle is sewn to another one, the particles state do not change. Nothing is to be done.

Simplifying: When a face that separates two volumes is removed, two cases arise. If the removed face doest not belong to any S_t state, nothing has to be done and the next PCD iteration will apply to the new merged volume. Otherwise, we replace the corresponding aimed at cell by an adjacent cell in the current polyhedron, the next PCD iteration will correct the prediction.

Cutting: This corresponds to a face insertion in a volume that splits it in two parts. All particles having a cell of the initial volume as a part of their S_t state are checked. If the face is between the particle and the aimed at cell then i_t is set to the inserted face. Otherwise the particle is still in the correct volume and nothing is to be done.

If during a geometrical deformation, the convexity constraint is lost for some cells, a local remeshing that uses the mentioned operation is performed.

To conclude, handling topological changes only requires updating the state of few particles that were lying inside the modified cell. It does not depend on the size of the scene and only involve local updates. The presented topological operations allow us to handle cutting, tying and ligaturing. Obviously other topological operations and deformations schemes can be treated similarly with only local updates.

6. Edge tests elimination

We propose two techniques in order to limit the overcost due to edge collision detection. The first improvement is based on the properties of the environment, the second one is based on kinetic information of the moving object. We propose an evaluation of this elimination techniques in section 7.

6.1. Elimination based on the environment

When an edge is totally included in a cell at a time t and its vertices are moved within the same cell, the edge cannot encounter any obstacles due to the convex property of the cells and thus do not need to be tested. When an edge was previously crossing a face of a cell of the environment or when a particle is crossing a face during a displacement, the edge has to be tested. The cost implied by this elimination is the memory cost, since the face crossing is already tested by the particle displacement algorithm. The performance of this elimination techniques is dependent of the relative size of the edges compared to the size of the cells of the environment.

6.2. Elimination based on the kinetic

Similarly to the back-face culling for collision detecion proposed in [Van94], it is possible to add a back-edge elimination test for closed polyhedron. Indeed, it is not necessary to test the edges going toward the *inside* part of the moving object. The inside of the object is assumed to be collision free at all times and thus do not require edge collision test. This edge tests can be removed by checking the direction of the displacement compared to the normal of the surrounding faces of a vertex. If we consider a mean valence of 6 for each edge and a minimal displacement cost of a 4 orientation tests for each particle displacement, this elimination test becomes profitable when at least 25% of the particles are moving toward the inside of the object. The worst case scenario for this elimination technique happens when each particle's velocity is tangent to the object, this happens seldomly. Most of the time, approximately 50% of the particles can be pruned.

6.3. Hybrid elimination

It is possible to combine the two elimination techniques. Three types of particles have to be taken into account : the particles whose edges displacements are crossing a face and going toward the outside of the object, the particles whose edge displacements are totally included within the cells of the environment and the particles crossing a face of the environment and going toward the inside of the object. This last type will be tested whenever they are moved toward the outside of the object.

7. Results and experimentations

We experiment our collision detection framework in medical and artificial environments with basic moving bodies to validate our method. We discuss here the efficiency of our framework and some implementation issues. Then we present a mass/spring simulation of a catheter moving in a tetrahedrized vascular network and an adaptation of the shape matching method introduced in [MHTG05]. We conclude our experimentations with an artificial test of the elimination techniques proposed.

7.1. Complexity

The core of our framework is the particle based collision detection that is used to throw particles along the trajectory of the vertices and along the moving edges. It has been shown in [JCD09] that if the displacements of the particles are small with respect to the sizes of cells, then checking if a particle encounters an obstable is achieved in $o(deg(c))$, where c is the cell that contains the particle and $deg(c)$ its degree (i.e. the number of faces for a volume).

This constraint with the size of displacement is compatible with usual physical simulations whose integration schemes have the same needs. This constraint also implies that the size of the edges of the moving bodies remains small enough. This can be achieved with adaptive subdivision. We show hereafter that this constraint may be removed in particular cases.

A key property of our framework is that the computation time of collision detection requests does not depend on the size of the scene, but only on the number of used particles. Using temporal coherence, stating that the aimed at cells sparsely change in a real simulation, an average of 7 point/plane orientation tests are sufficient to check collision for a particle. Considering the moving bodies are regular triangular meshes, we launch an average of 7 particles per vertex (1 for the trajectory and 6 for adjacent edges).

In classical approaches each trajectory is tested with a hierarchy of bounding boxes wrapping the entire scene, that gives time complexity in $o(log(n))$ for each particle, where n is the number of bounding box.

7.2. Mass/spring example

A mass/spring method is used to simulate the displacement of a catheter in a vascular network (figure 8). A distance

spring and an angular spring are used to model the catheter rigidity. Thanks to the edge collision detection it is possible to introduce momentum calculation when a collision arises on an edge. The physics of the catheter can be easily improved by using a constraint system [LCDN06] using the collision detection information for more realistic behavior.

For this example, we push a catheter inside the vascular network without adding particles. The length of the catheter, and therefore of the edges, increases during the simulation, invalidating the previously mentionned constraint. Therefore the number of tests increase with the size of the edges.

Figure 9 shows the number of tetrahedrons tested for collision as particles are spanned along edges. It increases with the length of edges. The spikes appearing on the graphic are the results of stress release in the simulation, which induce a bigger displacement of the catheter. As one can see, the number of tested tetrahedrons is kept small: below 3 percent of the total number of tetrahedrons.

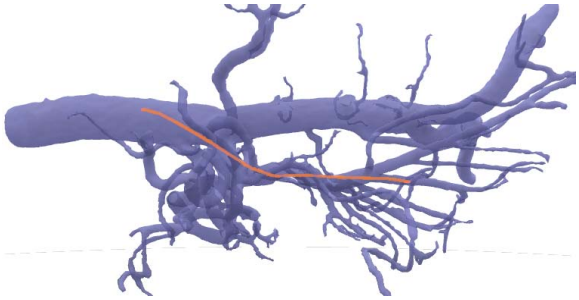

Figure 8: *A highlighted catheter moving in a vascular network. The catheter is sampled with 20 particles in a volumetric mesh of about 200k tetrahedrons.*

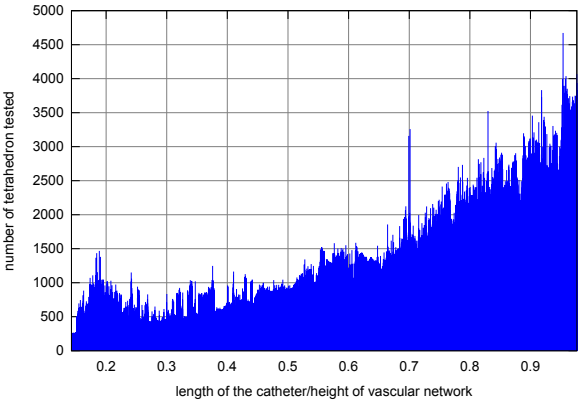

Figure 9: *Number of tetrahedrons tested in function of the length of the simulated catheter.*

7.3. Shape matching

We also adapt the shape-matching method to our framework. With this method, deformable bodies are represented as sets of particles without connectivity information. Their edges are only used for collision detection and do not support mechanical properties. The method is based on the computation of the minimal optimal linear transformation between a rest shape and a deformed state defined as a point cloud.

As expressed in equation 1 it consists in minimizing the error using the covariance matrix and in extracting the rotation part R between the rest shape and current state using Jacobian rotation. The translation T is considered as the vector between the center of the rest shape and the current one.

$$\begin{aligned} v_t^i &= p_t^i - (\sum_{i=1}^{N} p_t^i)/N \\ \varepsilon(R,T) &= \sum_{i=1}^{N} \|\omega_i(v_t^i - (Rv_0^i + T))\| \end{aligned} \tag{1}$$

In equation 1, v_t^i is one of the particles vector to the mass center from the set of N particles used to sample the object and v_0^i is the same particle vector when the object is in its resting shape. The parameter ω_i defines a parameter used to consider the particle significance in the shape which is detailed later.

7.3.1. Adaptive response

When using an adaptive response, the shape matching method has to be adapted. The resting position of the added point is interpolated as a weight-average of its neighborhood – i.e. the rest position of the subdivided edge.

$$v_0^a = r_{ToC} * v_0^i + vq_0^i(1 - r_{ToC}) \tag{2}$$

In equation 2, v_0^a is one of the added particles assumed resting vector, v_0^i and vq_0^i are the resting vectors of the particles of the edge in collision. r_{ToC} is the ratio representing the position of the collision compared to the edge length.

The parameter ω_i expressed in equation 1 is used to weight the new point significance. If too much weight is used for newly added particles, the transformation computation could result in an abnormal compensation toward them and induces unwanted rotation.

Whenever the computed transformation become negligible with non-colliding new added particles, the system automatically removes them.

7.4. Benchmark

Using the shape matching method, we simulated 225 deformable hexahedrons sampled with 8 particles during 1000

time steps with temporal coherence and with varying complexity of the scene. The hexahedrons are all initialized with a random impulse.

The environment is made of n volumetric hexahedrons with some fixed obstacles, we modify the number of hexahedrons to increase the complexity, as shown in figure 10. During the simulation random obstacles are added and removed from the scene.

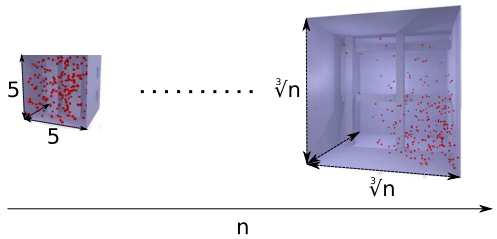

Figure 10: *Building environments of size n for the benchmark.*

The benchmark consists in comparing our algorithm with an AABB bounding volume hierarchy (BVH). The implementation of the AABB BVH is taken from the Bullet physic library. The proposed simulation has been implemented on a PC with a 2.4GHz Intel Core2 CPU and 2GB memory. Only one core has been used for the simulation.

Two tests have been performed with the BVH :

- one with segment/triangles intersection tests similar to the one presented for our edge collision detection without the use of a forecast mechanism
- one with triangle/triangle tests : each triangle is formed by : the moving vertex old position, the new position, and the position of one of the adjacent vertex. Bullet physic library use GIMPACT for triangle/triangle tests.

The time of the physical responses is ignored in both simulations. Figure 11 shows the numerical results. Our method overcome the others by a factor of 3. The segment/triangles intersection tests shows a clear logarithmic increasing with the complexity of the scene due to the underlying AABB BVH structure it use. The triangle/triangle test and our method are kept constant independently of the complexity. The triangle/triangle tests cost is amortized due to the objects repartition. Our method cost is kept constant because of the forecast mechanism used.

This benckmark did not make use of any of the elimination techniques described above.

We tested the elimination techniques previously described on an other scene shown in the accompanying video. The scene consists of a closed manifold hand mesh sampled with 12000 particles falling in an environment subdivided with pyramids, the time for collision detection is measured and represented on figure 12.

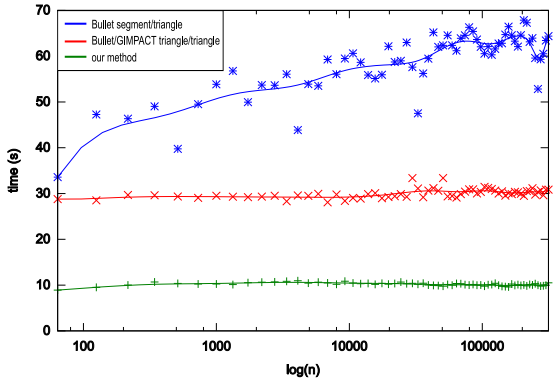

Figure 11: *Computation time for the collision detection in function of the complexity n of the scene for 225 deformable hexahedrons during 1000 time steps ; our method has a query time of about 10ms per time step meaning a time of approximately 5µs per particle.*

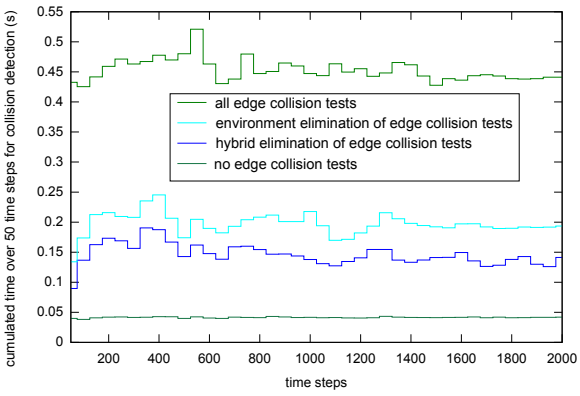

Figure 12: *Performance of the algorithm with different edge collision elimination techniques on the same closed manifold sampled with 12000 particles.*

As shown on the graphic, testing edge collision on this simulation increase the computation time by an order of magnitude of 9 compared to the use of PCD alone. By using the hybrid elimination technique, it is possible to lower this order of magnitude to 3. This improvement is dependent on the size of the cells, the size of the edges and the scale and the direction of the velocity.

8. Conclusion

In this paper we presented an edge collision detection method for deformable objects moving in a complex and deformable environment. The scene is modeled as a convex space subdivision and the bodies with manifold meshes.

This work relies on a particle-based collision detection

system. Particles are spanned along the trajectories and edges of the moving bodies to detect collision with obstacles in the environment. We show how our method accurately reports edge/environment contact information.

We presented effective use of this framework in two samples simulations: the insertion of a catheter in a vascular network and a shape matching extension supporting adaptive subdivision of deformable bodies. We present different kinds of responses allowed by our framework.

The ability of our method to efficiently handle topological changes allows us to simulate surgery operations. We plan to insert this work in a general framework dedicated to surgery simulations [ACF*07].

Finally, we compare our work with a standard library and demonstrate its efficiency. These good performances will allow us to experiment more sophisticated physical simulations in future works.

References

[ACF*07] ALLARD J., COTIN S., FAURE F., BENSOUSSAN P.-J., POYER F., DURIEZ C., DELINGETTE H., GRISONI L.: Sofa an open source framework for medical simulation. In *Medicine Meets Virtual Reality (MMVR'15)* (2007), pp. 1–6.

[BGTG03] BIELSER D., GLARDON P., TESCHNER M., GROSS M.: A state machine for real-time cutting of tetrahedral meshes. *11th Pacific Conf. on Comput. Graph. and Appl.* (2003), 377–386.

[BYM05] BELL N., YU Y., MUCHA P. J.: Particle-based simulation of granular materials. In *SCA : Proc. of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), pp. 77–86.

[CD99] CAZIER D., DUFOURD J.: A formal specification of geometric refinements. *Visual Comput. 15* (1999), 279–301.

[CLMP95] COHEN J. D., LIN M. C., MANOCHA D., PONAMGI M.: I-collide: an interactive and exact collision detection system for large-scale environments. In *Proc. of the symp. on Interactive 3D graphics* (1995), pp. 189–ff.

[CTM08] CURTIS S., TAMSTORF R., MANOCHA D.: Fast collision detection for deformable models using representative-triangles. In *Proc. of the symp. on Interactive 3D graphics and games* (2008), ACM, pp. 61–69.

[Duf89] DUFOURD J.-F.: Algebraic map-based topological kernel for polyhedron modellers: Algebraic specification and logic prototyping. In *Int. Conf. Eurographics'89* (1989), pp. 649–662.

[GD04] GUY S., DEBUNNE G.: *Monte-Carlo collision detection*. Tech. Rep. RR-5136, INRIA, 2004.

[Gei00] GEIGER B.: Real-time collision detection and response for complex environments. In *Proc. of the Int. Conf. on Comput. Graph.* (2000), pp. 105–113.

[GKJ*05] GOVINDARAJU N. K., KNOTT D., JAIN N., KABUL I., TAMSTORF R., GAYLE R., LIN M. C., MANOCHA D.: Interactive collision detection between deformable models using chromatic decomposition. *ACM Trans. Graph. 24*, 3 (2005), 991–999.

[GLM96] GOTTSCHALK S., LIN M. C., MANOCHA D.: Obbtree: a hierarchical structure for rapid interference detection. In *Proc. of the 23rd annual conf. on Comput. graph. and interactive techniques* (1996), pp. 171–180.

[Hub95] HUBBARD P. M.: Collision detection for interactive graphics applications. *IEEE Trans. on Visualization and Comput. Graph. 1*, 3 (1995), 218–230.

[JCA06] JOUSSEMET L., CROSNIER A., ANDRIOT C.: Espions: A novel algorithm based on evolution strategy for fast collision detection. In *EuroHaptics* (2006), pp. 159–167.

[JCD09] JUND T., CAZIER D., DUFOURD J.-F.: Particle-based forecast mechanism for continuous collision detection in deformable environments. In *SPM: SIAM/ACM Joint Conf. on Geometric and Physical Modeling* (2009), pp. 147–158.

[JFSO06] JIMÉNEZ J. J., FEITO F. R., SEGURA R. J., OGÁYAR C. J.: Particle oriented collision detection using simplicial coverings and tetra-trees. *Comp. Graph. Forum 25*, 1 (2006), 53–68.

[JH08] JANG H., HAN J.: Fast collision detection using the a-buffer. *Vis. Comput. 24*, 7 (2008), 659–667.

[KHM*98] KLOSOWSKI J. T., HELD M., MITCHELL J. S., SOWIZRAL H., ZIKAN K.: Efficient collision detection using bounding volume hierarchies of *k*-DOPs. *IEEE Trans. on Visualization and Comput. Graphics 4*, 1 (1998), 21–36.

[LCDN06] LENOIR J., COTIN S., DURIEZ C., NEUMANN P. F.: Interactive physically-based simulation of catheter and guidewire. *Comput. and Graph. 30*, 3 (2006), 416–422.

[Lie94] LIENHARDT P.: N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *Int. J. Comput. Geometry Appl. 4*, 3 (1994), 275–324.

[LM04] LIN M. C., MANOCHA D.: *Handbook of Discrete and Computational Geometry*. 2004, ch. 35 - Collision and proximity queries.

[LT07] LINDBLAD A., TURKIYYAH G.: A physically-based framework for real-time haptic cutting and interaction with 3d continuum models. In *Proc. of the ACM symp. on Solid and physical modeling* (2007), pp. 421–429.

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (2005), 471–478.

[OL03] OTADUY M. A., LIN M. C.: Clods: dual hierarchies for multiresolution collision detection. In *Proc. of the Eurographics/ACM SIGGRAPH symp. on Geometry processing* (2003), pp. 94–101.

[TCYM09] TANG M., CURTIS S., YOON S., MANOCHA D.: Iccd: Interactive continuous collision detection between deformable models using connectivity-based culling. In *IEEE Trans. on Visualization and Comp. Graph.* (2009), pp. 544–557.

[THM*03] TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANERTS D., GROSS M.: Optimized spatial hashing for collision detection of deformable objects. In *Vision, Modeling, Visualization* (2003), pp. 47–54.

[TKZ*04] TESCHNER M., KIMMERLE S., ZACHMANN G., HEIDELBERGER B., RAGHUPATHI L., FUHRMANN A., CANI M.-P., FAURE F., MAGNETAT-THALMANN N., STRASSER W.: Collision detection for deformable objects. In *Eurographics State-of-the-Art Report* (2004), pp. 119–139.

[TMT10] TANG M., MANOCHA D., TONG R.: Fast continuous collision detection using deforming non-penetration filters. In *ACM SIGGRAPH Symp. on Interactive 3D Graphics and Games* (2010).

[Van94] VANECEK G.: Back-face culling applied to collision detection of polyhedra. *Journal of Visualization and Computer Animation volume 5*, 1 (1994), 55–63.

[vdB97] VAN DEN BERGEN G.: Efficient collision detection of complex deformable models using aabb trees. *J. Graph. Tools 2*, 4 (1997), 1–13.