Revue Électronique Francophone d'Informatique Graphique, Volume 3, Numéro 2, pp. 47–58, 2009

Système de prédiction pour la détection de collisions dans un environnement déformable

Thomas Jund [1] David Cazier [1] et Jean-François Dufourd [1]

[1] Université de Strasbourg
LSIIT, UMR CNRS 7005

Résumé

La détection de collisions entre un mobile et son environnement est cruciale dans le contexte des applications de simulation physique. Beaucoup de méthodes ont été proposées pour accélérer cette détection dans une scène complexe en vue de respecter la contrainte de l'interactivité. Cependant, lorsque l'environnement devient déformable, les optimisations basées sur l'utilisation de structures hiérarchiques trouvent leurs limites au niveau du temps de mise à jour. Nous proposons dans cet article une méthode basée sur une subdivision volumique de l'environnement couplée à un mécanisme de prédiction. L'environnement est subdivisé en un ensemble de cellules convexes enrichi par une structure topologique forte permettant des requêtes de voisinages optimales. Ce système de prédiction utilise la cohérence temporelle et les propriétés des cellules pour optimiser le nombre de tests d'intersection à effectuer à chaque pas de temps. Nous illustrons notre méthode avec l'animation de flots de particules dans des maillages déformables.

Mots clé : Détection de collisions, cartes combinatoires, environnement déformable, simulation physique, système de particules.

1. Introduction

La détection de collisions est un problème important dans la plupart des simulateurs physiques. Ce type d'application nécessite une interactivité complète et des taux de rafraîchissement élevés pour permettre des rendus réalistes ou encore pour le couplage avec des systèmes à retour d'effort. Le temps de calcul requis pour la détection de collisions doit donc être minimal pour assurer la meilleure interactivité possible. Tout en respectant ces contraintes, les opérations de détection doivent être robustes et consistantes, sans quoi les informations renvoyées par le simulateur seraient faussées et le réalisme serait perdu.

Un des domaines d'applications que nous visons est la simulation chirurgicale, un domaine très contraignant pour la détection de collisions. Dans ce type d'applications les objets sont déformables et doivent être capables de supporter des changements de topologie (pour inciser un organe par exemple). Par contre, l'environnement géométrique dans lequel se déroule la simulation (dans ce cas, le corps humain) ainsi que les outils utilisés sont connus à l'avance et subissent peu de changements.

Il est usuel de représenter l'environnement (par exemple : un réseau vasculaire, une trachée ou le colon) par une approche à deux niveaux. Un niveau grossier définit la structure et est utilisé pour la détection de collisions. Le deuxième niveau contient les triangles associés à la première structure, représentant la géométrie de l'environnement et est utilisé pour le rendu. Notre proposition est de permettre d'unir ces deux niveaux dans une même structure volumique, évitant ainsi les problèmes de mise à jour lors de changements topologiques.

Notre approche consiste à subdiviser l'environnement en cellules convexes et à suivre la trajectoire de l'outil à l'intérieur de cette subdivision. L'utilisation d'une structure topologique nous permet d'atteindre une complexité en temps constant pour les requêtes de voisinage. Nous utilisons cette subdivision pour optimiser le suivi en prédisant, pour chaque déplacement, la cellule que l'outil est sur le point d'atteindre.

Cet algorithme est basé sur la cohérence temporelle et spatiale. Pour chaque point du mobile nous mémorisons la dernière position et la dernière prédiction effectuée. Grâce à ce mécanisme, nous n'avons qu'à calculer un mouvement minimal pour chaque déplacement du mobile.

Nous partons de la supposition que le taux de rafraîchissement du mobile est suffisamment élevé pour considérer une trajectoire linéaire. Autrement nous devrions interpoler

la trajectoire à partir de plus de positions, et donc augmenter la complexité de nos tests. Nous supposons également que l'objet en mouvement peut être discrétisé un en ensemble de points (ou de petites sphères). Ces suppositions sont communément acceptées dans les simulations dites *meshless*.

Le déplacement du mobile est considéré comme externe à notre algorithme. Les réactions physiques, telles que la gravité ou les frottements, ou encore les changements de direction sont cependant supposées être cohérentes. Si le déplacement du mobile est estimé comme étant *chaotique*, c'est à dire que la trajectoire entre deux pas de temps ne peux être considérée comme linéaire, l'utilisation de la cohérence temporelle s'avère inutile.

Des réponses physiques robustes, telles que des calculs de forces ou des changements de topologie de type fusion ou découpage sont importantes dans le cadre de la simulation chirurgicale. Pour une question de concision, nous nous focaliserons principalement sur l'algorithme de détection de collisions.

Bien que la simulation 3D soit notre objectif principal, nous présentons ici également le cas de la dimension 2 pour clarifier la méthode. Ainsi notre exposé se découpe comme suit. Dans la section 2, nous rappelons les travaux en détection de collisions. Dans la section 3, nous présentons la décomposition de notre environnement en un ensemble de cellules convexes. Dans la section 4, nous introduisons le principe utilisé pour la navigation en dimension 2. Dans la section 5, nous présentons notre algorithme en dimension 3. Nous terminons par un exemple de l'utilisation de notre algorithme dans un environnement déformable et par des perspectives.

2. État de l'art

Nous présentons ici un bref état de l'art du domaine de la détection de collisions. Seules les méthodes les plus représentatives sont citées ci-dessous. Pour un état de l'art plus complet on pourra se référer à [JTT01, Jou06, LM04, TKZ*04, PB07].

Une méthode répandue consiste à utiliser des boites englobantes pour réduire la complexité des tests d'intersections. Ces tests, effectués sur des polyèdres simples, comme des AABB (Axis-aligned bounding box) [CLMP95], OBB (Oriented Bounding Box) [GLM96] ou *k*-dop (K Discrete Oriented Polytope) [KHM*98], permettent d'éliminer des collisions éventuelles de la scène sans avoir à effectuer de tests précis. L'inconvénient de ces méthodes se situe au niveau de la mise à jour de la structure lorsque l'on considère des objets déformables.

Une approche entièrement différente consiste à rechercher les points les plus proches entre deux objets afin de tester s'il y a, ou non, collision. On citera notamment ici les algorithmes GJK et Lin-Canny [GJK88, LC91]. Ce type d'algorithme ne s'utilise que sur des objets convexes ou sur une décomposition convexe mais n'est pas particulièrement adapté aux environnements déformables complexes, car aucune modification topologique n'est supportée.

Une autre approche intéressante, utilisée en simulation, est l'utilisation de la multirésolution. Dans [OL03], un système appelé *CLOD*s – Contact Level Of Details – est décrit. Le principe est le suivant : si une collision a lieu le long d'une surface de contact large, alors cette surface peut être considérée à une résolution plus faible. Un autre concept lié aux CLOD qui a été introduit par Hubbard dans [Hub95] est le principe de temps critique. Le calcul est fait à la résolution la plus précise atteignable dans un temps limite fixé.

Une autre manière de diminuer le nombre de tests est l'utilisation d'une approche de type Monte-Carlo. Par le biais d'une approche stochastique [GD04] ou évolutionniste [Jou06], ces méthodes sélectionnent un échantillonnage sur les objets à tester et utilisent la cohérence temporelle pour faire tendre cet échantillonnage vers les points de proximité entre les objets.

Certains travaux, plus proches des applications que nous visons, s'attellent à traiter le cas d'objets naviguant dans de grands environnements, comme la simulation d'endoscopie ou d'angioscopie [Gei00, Aco04, LCDN06, WDS*07]. Dans ces approches les mobiles sont souvent modélisés comme un ensemble discret de points ou de segments selon la précision requise pour le simulateur.

Les approches de [Aco04, WDS*07] utilisent une hiérarchisation spatiale ou une carte de distance pour détecter la collision. Avec ce type de structure, une requête de collision a une complexité de l'ordre de $O(log n)$ où n est la taille de la structure. De plus, ces modèles ne supportent pas facilement les modifications de topologie ou les déformations.

La méthode de [LCDN06] est basée sur un arbre de cellules, chacune des branches représentant une partie du vaisseau. Cette approche est particulièrement adaptée pour la navigation dans des vaisseaux sanguins. Mais de même que précédemment, cette méthode ne prend pas en compte des éventuels changements de topologie. En outre, cette méthode nécessite un deuxième niveau de détails pour représenter la paroi des vaisseaux.

Les travaux de [Gei00] sont basés sur une décomposition spatiale de l'espace en tétraèdres, et sur l'utilisation de la cohérence temporelle. Cette décomposition permet l'utilisation de l'algorithme sur des structures non tubulaires. L'inconvénient de cette méthode est le manque de structure topologique. Ce manque se traduit par une restriction des modifications possibles sur l'environnement en temps interactif. De plus, tous les tétraèdres sont exprimés de manière explicite dans le modèle. Notre méthode, se basant aussi sur des tétraèdres, ne les utilise que de manière implicite, tel que montré ci-après, et nécessite donc de stocker moins d'informations.

3. Modélisation de l'environnement

3.1. Modèle topologique

Le mécanisme de prédiction que nous proposons est basé sur une subdivision de l'environnement dans lequel les collisions sont possibles. Le mouvement des mobiles est contraint dans l'espace partitionné en cellules (sommets, arêtes, faces et volumes). Beaucoup de structures existent en modélisation géométrique pour représenter ce types de subdivisions en conservant les relations d'adjacences [Bau75, Bri93]. Parmi celles-ci, nous choisissons les cartes combinatoires, qui sont une formalisation de ces structures de données [Tut84, Lie91]. L'intérêt de ce type de structure topologique a déjà été illustré dans beaucoup de travaux [GS85, EG86, GHPT89, CD99].

Notre méthode est similaire à celle de [LCDN06] car elle utilise les relations d'adjacences entre les différentes cellules sans utiliser de structure d'arbre. Elle est cependant plus générale dans le sens où elle s'applique à des environnements contenant n'importe quel type de structure interne, déformables et acceptant les changements de topologie. De plus, les cartes combinatoires permettent l'utilisation de maillage multirésolution [KCB08] pour la visualisation ou encore l'adaptativité de la précision. Elles présentent aussi une définition formelle et mathématique qui permet de traiter le problème de la robustesse. Des modèles topologiques similaires ont déjà été utilisés pour traiter des opérations spécifiques telles que la suture ou l'incision [BGTG03, LT07].

Une *carte* de dimension n est un ensemble fini de *brins* équipé de n permutations $\phi_1, \ldots, \phi_n$. Habituellement un brin est représenté par une demi-arête orientée. Les relations ϕ_i sont utilisés pour coudre les brins les uns aux autres et former des i-cellules. De plus, la relation ϕ_1 est une permutation et $\phi_2, \ldots, \phi_n$ sont des involutions (i.e. une permutation telle que $\phi_i(\phi_i(x)) = x$). Ces relations doivent vérifier que $\phi_i \circ \phi_j$ est une involution pour $i \geq 1$ et $j > i+1$. Ces contraintes garantissent que la subdivision construite représente une partition bien formée d'une variété fermée de dimension n [Lie91].

Dans des termes plus concret, un brin x est dit cousu en i à un brin y quand $\phi_i(x) = y$. Les brins cousus selon ϕ_1 forment un cycle d'arêtes orientées ou *face*. Lorsque deux brins sont cousus selon la relation ϕ_2, leurs deux faces sont cousues le long des arêtes orientées x et y. En dimension 2, une 2-carte modélise une partition du plan en faces cousues par ϕ_2.

La figure 1 montre les détails d'une 2-carte contenant 4 faces. Le brin x représente une arête orientée. Le brin $\phi_1(x)$ est le brin qui suit x dans la face. Le brin x représente également la face à laquelle il appartient (comprenant $\phi_1(x)$, $\phi_1(\phi_1(x))$, etc...). Le brin $\phi_2(x)$ est le brin de la même arête que x mais de la face adjacente. Par définition, $\alpha_1(x)$ est $\phi_1 \circ \phi_2(x)$ et est le brin suivant x dans le sommet de x.

En dimension 3, un ensemble de faces cousues par des liaisons ϕ_2 forme une surface fermée et orientable qui définit

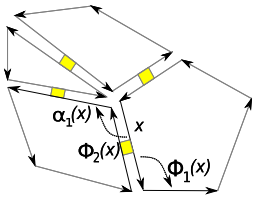

Figure 1: *Détails d'une 2-carte. Un cycle de brins forme une face. Une couture en ϕ_2 entre deux brins, représentée par un rectangle jaune, se traduit par une relation d'adjacence entre les deux faces dont ils font partie.*

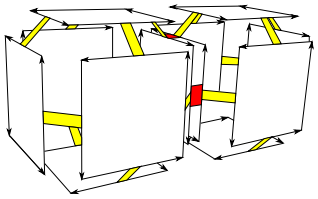

Figure 2: *La vue explosée d'une 3-carte (duale) de deux cubes cousus : une flèche représente un brin, deux flèches consécutives sont liées par ϕ_1 ; un rectangle jaune entre deux brins symbolise une liaison ϕ_2 ; un rectangle large rouge entre deux brins symbolise une liaison ϕ_3.*

une 3-cellule, c'est-à-dire un volume orienté. Les volumes sont cousus entre eux par des liaisons ϕ_3. La condition mentionnée précédemment, stipulant que $\phi_1 \circ \phi_3$ est une involution, contraint les volumes à toujours être cousus le long de faces entières. Une 3-carte modélise une partition d'un objet 3D en volumes cousus par ϕ_3 (figure 2).

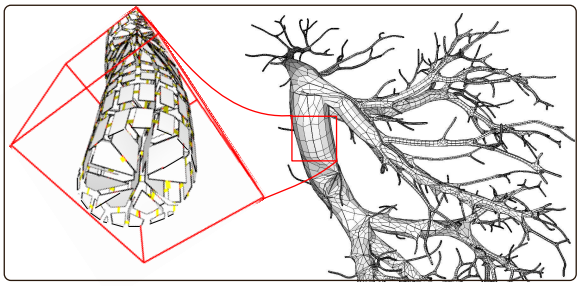

Figure 3: *Un réseau vasculaire.*

3.2. Contraintes géométrique

Notre méthode, qui veille à réduire au maximum le nombre de tests d'intersection nécessite une décomposition de l'environnement en cellules convexes. Ainsi en dimension 2, nous considérons une partition du plan en faces convexes. En dimension 3, la scène est une subdivision en polyèdres convexes tels que chacune de leurs faces soit

convexe (figure 3). Nous supposons également que la décomposition cellulaire ne contient pas d'arête ou de face dégénérée.

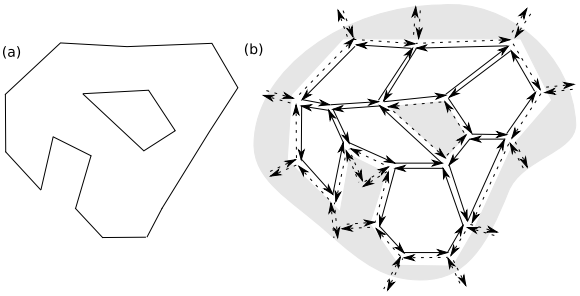

Figure 4: *Un environnement 2D et sa décomposition cellulaire ; les cellules infranchissables sont en pointillées.*

Le mobile peut se déplacer librement dans les cellules qui sont dites *libres*. A contrario, les autres cellules correspondant au bord extérieur ou à des obstacles intérieurs sont infranchissables. Des marqueurs, associés aux brins, sont utilisés pour signaler ces zones (figure 4(b)).
Il faut noter que la face ou le volume extérieur n'est jamais convexe. Pour que notre méthode fonctionne en dimension 2, nous ajoutons alors des arêtes et des faces fictives qui rendent l'extérieur convexe (la figure 4(b) en illustre l'idée). En dimension 3, nous englobons notre scène dans un volume externe subdivisé afin d'assurer le respect de la contrainte de convexité.

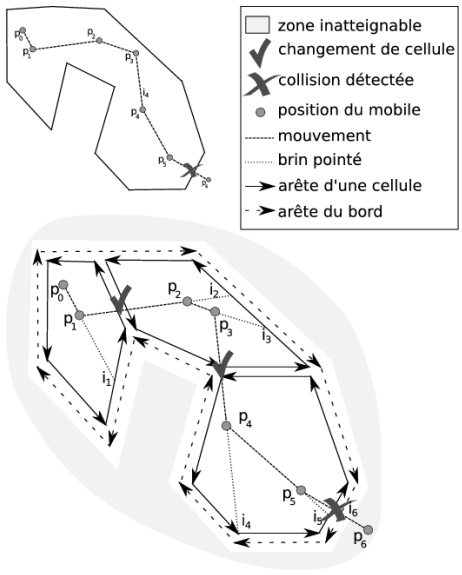

Figure 5: *En haut à gauche : le déplacement d'un mobile dans un environnement 2D. En bas, la décomposition de l'environnement et donc du mouvement sur des cellules convexes.*

4. Navigation 2D

Le problème que l'on cherche à résoudre est de savoir si le mobile entre en collision avec le bord de l'environnement lorsqu'il passe d'une position p_t à une position p_{t+1}. La méthode naïve consisterait à tester l'intersection du segment $[p_t, p_{t+1}]$ avec l'ensemble des arêtes du bord. La décomposition en cellules convexes permet de limiter ces tests aux arêtes du bord de la cellule pour peu que l'on sache à tout moment dans quelle cellule on se trouve. L'idée que nous défendons est que si l'on sait vers quelle arête se déplace le mobile, alors un seul test d'intersection est nécessaire. Notre algorithme consiste donc à prédire, le mieux possible, l'arête vers laquelle le mobile se dirige, en notant que lorsqu'il suit une trajectoire régulière cette arête change rarement. Nous suivons les changements de trajectoires avec un nombre de tests minimal, en prenant appui sur la cohérence temporelle.

4.1. Vue générale

À un instant t, l'état d'un mobile est donné par sa position p_t, le type k_t de la cellule le contenant (face, arête, ou sommet) et un brin i_t de cette cellule, nous appelons S_t ce triplet. Au temps $t+1$, le point p_t est déplacé selon une trajectoire, que l'on suppose rectiligne, jusqu'à un point p_{t+1}. Notre algorithme consiste à chercher le type k_{t+1} de la cellule contenant p_{t+1} ainsi que le brin i_{t+1} coupé par la demi-droite $[p_t, p_{t+1})$. Ce brin indique la localisation d'une éventuelle collision.

Lorsque le mobile avance jusqu'à la position p_{t+1}, notre algorithme cherche i_{t+1} et k_{t+1} en se basant sur le triplet S_t et sur p_{t+1}.

Une fois cette prédiction effectuée, une collision peut arriver si et seulement si il y a intersection entre $[p_t, p_{t+1}]$ et l'arête correspondante à i_{t+1}. La figure 5 illustre ce principe.

Afin de trouver S_{t+1}, plusieurs cellules intermédiaires peuvent être traversées. Plusieurs types k'_{t+1} de cellules et plusieurs brins i'_{t+1} intermédiaires sont alors utilisés jusqu'à trouver le bon couple (k_{t+1}, i_{t+1}) (figure 6). La cellule de la prédiction est trouvée lorsque i'_{t+1} nous indique que le point p_{t+1} ne change pas de cellule, ou qu'il y a eu une collision.

Le brin i'_{t+1} nous permet de savoir s'il y a eu collision, changement d'état, ou encore si nous avons trouvé l'état S_{t+1}. Si le point p_{t+1} reste dans la cellule de type k'_t alors on a trouvé le bon état (figure 7). Sinon soit le franchissement est interdit et nous reportons alors une collision, soit nous devons passer au type k'_{t+1} de cellule correspondant au franchissement de cet intervalle puis chercher le bon i'_{t+1} de ce dernier.

4.2. Mécanisme de prédiction

Nous présentons ici, pour chacun des états atomiques de notre mécanisme de prédiction, comment prévoir de manière

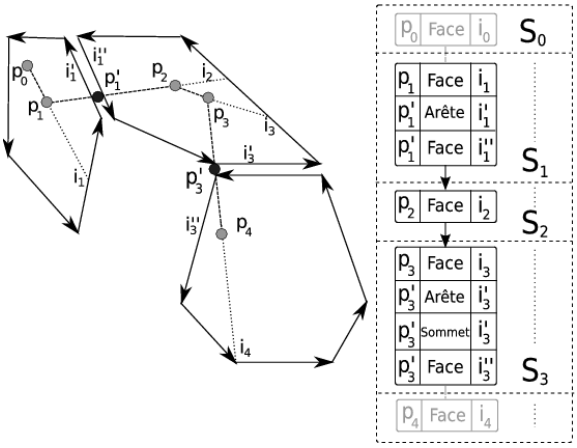

Figure 6: *Zoom sur le principe. On voit la décomposition de la recherche du triplet S_{t+1} à partir du triplet S_t. Il est parfois nécessaire de passer par différents types de cellules et de brins de prédictions avant d'arriver à l'état cherché.*

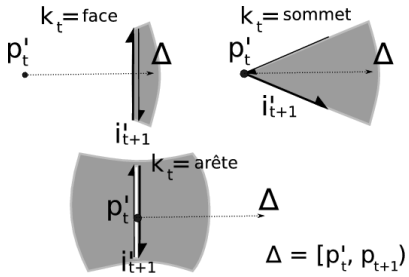

Figure 7: *Cas de changements d'états. Pour chacun des états le brin i_{t+1} correspond à la prédiction selon la direction du mouvement. Dans une face : on change d'état si on franchit le brin. Sur une arête : on change d'état si on quitte l'arête. Sur un sommet : on change d'état dès que l'on sort du sommet.*

optimal l'évolution de l'état courant S_t. Dans tous les cas, la prédiction est composée d'une phase d'orientation puis d'une phase de déplacement.

Le mouvement du mobile de p_t vers p_{t+1} est décomposé en déplacements élémentaires correspondant aux changements d'états et de cellules. Pour chacun de ces déplacements nous notons s le point source, c le point cible et d le brin correspondant à la prédiction actuelle. Au début du pas de temps $[t,t+1]$ on a $s = p_t$, $c = p_{t+1}$ et $d = i_t$. Nous notons enfin d_1 et d_2 les sommets de l'arête d (figure 8)) et Δ la demi-droite $[s,c)$.

Pour les faces, la phase d'orientation consiste à tourner dans la face autour du point s (figure 8), en utilisant l'opération ϕ_1, pour trouver le brin dont l'arête est coupée par la droite Δ ; pour optimiser les calculs nous effectuons des

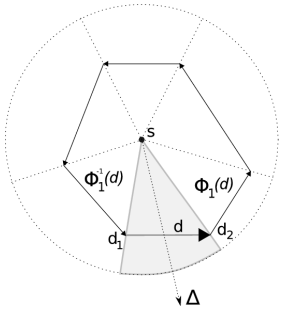

Figure 8: *Orientation sur une face. Le cycle d'arêtes de la face définit une partition en secteurs angulaires.*

tests d'orientation, le brin est trouvé lorsque c est à droite de $[s,d_2)$ et à gauche de $[s,d_1)$ –sur la figure 8 il s'agit de l'intervalle en couleur–. Selon la réponse du test d'orientation, nous nous déplaçons au brin suivant de la face – $\phi_1(d)$ – ou au brin précédent de cette dernière – $\phi_1^{-1}(d)$ – pour trouver le bon intervalle. En utilisant cette méthode le nombre de brins testés est minimal.

Nous avons réduit le nombre de brins testés au minimum mais pas encore le nombre de tests pour chacun des brins. Afin d'être minimal dans le nombre de tests d'orientation nous réutilisons en permanence les résultats des tests précédents. Le principe est de commencer la rotation sur la face dans le sens où l'angle vers c est le plus faible puis de continuer dans cette direction jusqu'à trouver la réponse attendue, c'est-à-dire jusqu'à ce que le bon intervalle soit trouvé.

La phase de déplacement est plus aisée, elle consiste à vérifier s'il y a, ou non, intersection entre $[s,c]$ et d –toujours en utilisant un test d'orientation–. S'il n'y a pas d'intersection, on renvoie l'état courant, sinon nous remplaçons le point s par le point d'intersection et nous continuons la prédiction dans l'état arête.

L'état sommet est le dual de l'état face : la source est le sommet, les intervalles considérés sont définis par les arêtes (figure 9). L'algorithme pour les sommets est similaire à celui décrit pour les faces. La phase de déplacement consiste à vérifier si c se trouve sur une arête de l'intervalle ou dans la face.

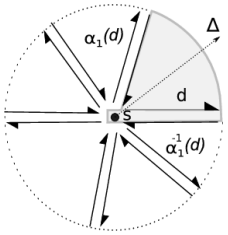

Figure 9: *Phase d'orientation des sommets.*

Le traitement des arêtes se résume à 5 possibilités. Soit le point se trouve sur une des faces adjacentes, soit sur l'arête, soit il a franchi un des deux sommets. Si le point c est dans une face, on teste alors si la face est atteignable ou marquée. Si le point a franchi un sommet on va dans l'état sommet correspondant sinon on reste dans l'état arête.

4.2.1. Complexité

Nous évaluons la complexité de chaque état atomique séparément. La complexité entre un état S_t et un état S_{t+1} est fortement corrélé à la taille et la direction du déplacement effectué. La complexité maximum est également dépendante du *stabbing number* [HKM95]. Ce nombre est le nombre maximum de cellules que peut traverser un rayon avant de rencontrer un obstacle. Comme l'algorithme se base sur des tests d'orientation, la complexité est exprimée selon le nombre de tests nécessaires pour chaque type de cellule. Afin de connaître la complexité d'une situation précise, la complexité devrait être évaluée en considérant la taille et la géométrie des cellules tout en considérant la taille maximale d'un déplacement que l'on peut effectuer lors d'un pas de temps.

La complexité lorsque la cellule considérée est une face est en $n-1$ dans le pire des cas, pour une face composée de n arêtes, i.e. lorsque l'objet fait un demi-tour ou lorsqu'il traverse une arête. Dans la plupart des cas, l'outil d'interaction suit une trajectoire rectiligne : dans ce cas la complexité est constante ; on vérifie juste si le brin visé reste le même et s'il coupe une arête.

La complexité lorsque la cellule considérée est une arête est en temps constant. Il faut au plus 3 tests pour séparer les 5 cas présentés.

Pour le cas d'un sommet incident à n faces, la complexité de l'orientation peut être de $n-1$ dans le pire des cas, comme celle de l'état face. Dans la plupart des cas, on peut supposer que la répartition des faces autour d'un sommet est régulière, ainsi la complexité (moyenne) approche les $n/2$.

5. Navigation 3D

La navigation 3D suit le même principe que celui de la navigation 2D. On décompose la navigation en 4 états : volume, face, arête et sommet. A chaque pas de temps, on cherche toujours le brin i_{t+1} en se basant sur l'état précédent. La complexité de l'algorithme augmente car la prédiction se fait en 3 dimensions.

5.1. Prédiction dans l'état volume

Lorsque le mobile se trouve dans un volume, la prédiction consiste maintenant à trouver le tétraèdre formé par : le brin courant, le point s et un point sur la face que nous appelons s_p, qui est une approximation de la projection de notre point c sur la face que nous sommes en train de tester (figure 10).

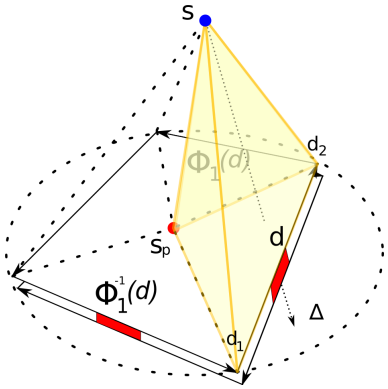

Figure 10: *Phase d'orientation de l'état volume ; seule une face du volume est montrée pour une question de lisibilité.*

Durant la phase d'orientation de l'état volume, nous cherchons le tétraèdre contenant Δ (figure 10). Nous sommes bien orienté lorsque c est inclus dans le dièdre formé par les plans $(s, s_p, d2)$ et $(s, s_p, d1)$ (l'intervalle central représenté sur la figure 10). Comme nous considérons ici le cas 3D, le point peut se trouver au-dessus du plan $(s, d1, d2)$, dans ce cas, on recommence la phase d'orientation sur la face adjacente à la face courante par rapport à ce dernier plan.

Les points s et s_p doivent être respectivement à l'intérieur du volume et sur la face que nous sommes en train de tester pendant la recherche du tétraèdre afin de couvrir intégralement l'espace. Lorsque nous changeons de face à l'intérieur d'un volume, nous calculons un nouveau point s_p arbitrairement à l'intérieur de la nouvelle face pour respecter cette contrainte.

Une fois que nous avons trouvé le bon tétraèdre il ne reste plus qu'à tester si le segment $[sc]$ coupe la face se trouvant à la base du tétraèdre. En cas d'intersection, le mobile passe à l'état face et le point s est placé au point d'intersection, autrement la prédiction est terminée.

5.2. Prédiction dans l'état face

Lorsque le mobile est dans l'état face, nous réutilisons le principe présenté dans l'algorithme 2D. Ici, les tests d'orientation sont basés sur les plans formés par : le brin courant, le point s et le point s translaté le long de la normale à la face pour trouver le bon triangle – afin d'éviter de calculer des projections –. La prédiction dans l'état face est similaire à celle du cas 2D excepté pour une particularité. Le point c peut quitter la face en se déplaçant vers le haut ou vers le bas (figure 11). Pour traiter cette particularité, nous vérifions simplement si le volume dans lequel c se dirige est atteignable et signalons une collision ou continuons dans l'état volume selon le cas. Lorsque le volume est atteignable, nous plaçons s_p sur la première face non planaire à celle depuis

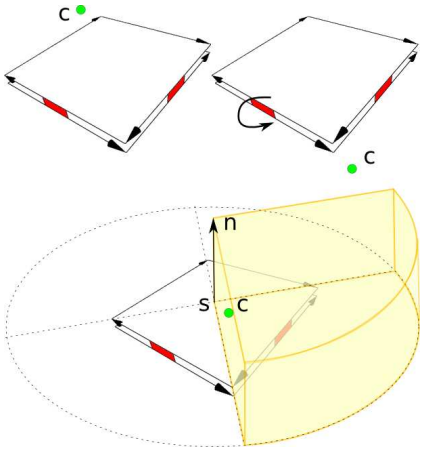

Figure 11: *Phase d'orientation de l'état face. On teste d'abord si c est au-dessus ou sous la face. Si ce n'est pas le cas, on tourne autour de la face pour trouver le bon secteur angulaire.*

laquelle on vient, afin de ne pas avoir de tétraèdre plat pour continuer notre prédiction.

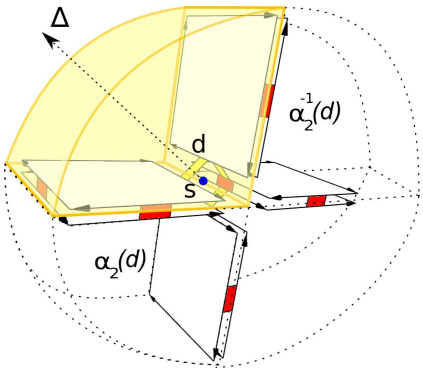

Figure 12: *Phase d'orientation de l'état arête. On tourne autour de l'arête en cherchant le secteur angulaire, défini par deux plans, et contenant* Δ.

5.3. Prédiction dans l'état arête

Durant la phase d'orientation de l'état arête, nous cherchons l'intervalle défini par un volume contenant la demi-droite Δ en utilisant des tests d'orientation (figure 12). Une fois cet intervalle trouvé, la phase de déplacement cherche à identifier si c est sur l'arête, dans un volume, ou sur une face. Si le point est sur l'arête nous vérifions si le point est sur $]d_1, d_2[$ ou s'il a atteint un sommet. Après ces tests, nous signalons si le point est sur l'arête ou continuons la prédiction dans l'état trouvé.

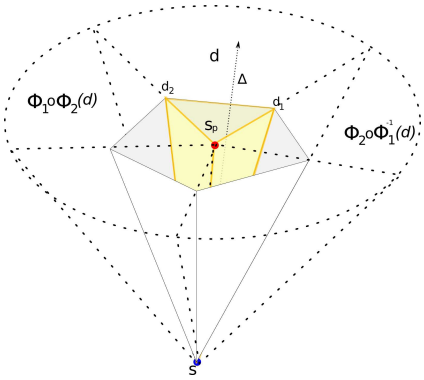

Figure 13: *Phase d'orientation de l'état sommet. On tourne dans l'ombrelle qui est décomposée en secteurs angulaires avant de vérifier si on est bien dans le bon volume ; jusqu'à trouver le bon secteur angulaire dans le bon volume.*

5.4. Prédiction dans l'état sommet

La prédiction de l'état sommet est le dual de celle du volume. Comme il n'est pas possible d'utiliser un point s_p sur une des faces, nous choisissons un point à l'intérieur de la partition de l'espace définie par l'intersection des demi-espaces que forment les faces (figure 13). Nous cherchons le bon tétraèdre en nous basant sur ce point. Une fois trouvé, le dual du test avec (s, d_1, d_2) consiste à tester si l'on est dans le volume actuel ou le volume adjacent. A la fin des tests d'orientation, le bon tétraèdre à été trouvé et il est possible de prédire le nouvel état selon la position de c le tétraèdre.

5.5. Complexité

De la même manière que la complexité du cas 2D, nous exprimons la complexité selon le nombre de tests d'orientation. La complexité des tests lorsque le type de cellule est un volume est en $n-1$ dans le pire des cas, pour un volume composé de n faces. Si la distribution des faces dans le volume est régulière, on approche plutôt une complexité (moyenne) en $n/2$.

Lorsque la cellule considérée est de type face, la complexité est la même que celle du cas 2D en dehors du test consistant à vérifier l'appartenance à la face.

La complexité des tests pour une arête en 3D est la même que celle des sommets dans le cas 2D.

La complexité des tests pour un sommet est la même que celle des volumes, étant donné qu'il s'agit du cas dual.

Dans la plupart des cas, si le mobile ne change pas de direction et n'entre pas en collision, le nombre de tests est minimal et correspond aux tests d'orientation avec les 4 plans du tétraèdre courant.

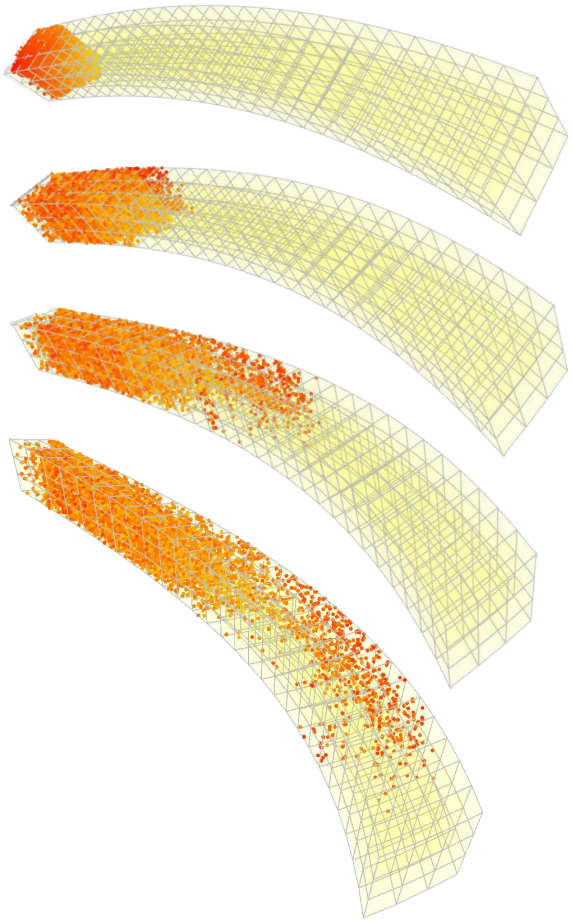

Figure 14: *10 000 points dans une maillage en déformation.*

point p_{t+1}. Le point p_m ne doit pas nécessairement être le barycentre de la cellule, tout point contenu dans la cellule précédente convient pour l'algorithme.

En pratique, si la déformation permet de déduire la position actuelle de p_t et p_{t+1} par rapport à la cellule, on évaluera les deux premiers cas. Dans le cas contraire, n tests d'orientation sont nécessaires pour obtenir cette information. Dans ce cas, on peut éventuellement directement considérer le troisième cas sans effectuer ces tests.

Pour un maillage de 3x3x26 cellules cubiques tel que présenté sur la figure 14, nous avons effectué 2 expérimentations pour valider notre approche. Nous avons lâché 1000 particules soumises à la gravité pendant 1000 pas de temps. Cette expérience a été effectuée dans le cas d'un maillage statique, et d'un maillage déformable.

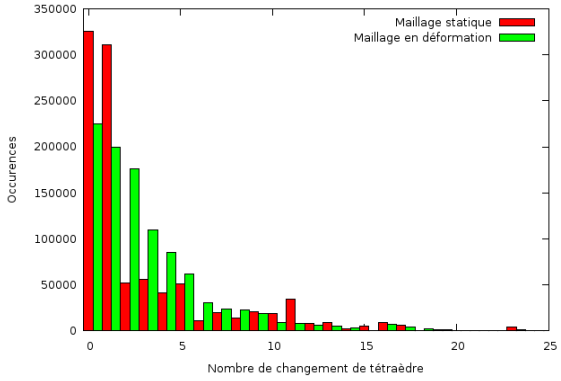

Figure 15: *Nombre de mise à jour nécessaire de tétraèdre pour 1000 particules pendant 1000 pas de temps dans le cas d'un maillage statique et d'un maillage déformable.*

6. Expérimentations

6.1. Détection de collisions dans un maillage déformable

Comme le montre la figure 14, l'utilisation de la topologie pour la détection de collisions permet à notre algorithme de fonctionner sur un maillage déformable sans avoir à effectuer de mise à jour coûteuse.

Lorsque le maillage est déformé, il nous faut vérifier que la prédiction est toujours valide. Trois cas surviennent :

- Cas 1 : p_t et p_{t+1} sont toujours dans la même cellule et p_{t+1} se trouve toujours dans le tétraèdre de prédiction. La prédiction est toujours valide.
- Cas 2 : p_t est toujours dans la même cellule mais le tétraèdre de prédiction n'est plus valide. On effectue une prédiction sans déplacer les points pour la corriger.
- Cas 3 : p_t n'est plus dans la même cellule. Ce cas intervient lorsque l'enveloppe de la cellule passe sur p_t. Dans ce cas, pour trouver la nouvelle cellule contenant p_{t+1}, nous simulons un déplacement d'un point p_m se trouvant à l'intérieur de la cellule précédente jusqu'au

Comme le montre la figure 15, on voit que la prédiction change rarement dans les deux cas. Le cas du maillage déformable subit quelques changements de tétraèdre supplémentaire. En terme de nombre de tests d'orientation, le cas statique nécessite en moyenne 6.75 tests d'orientation alors que le cas déformable en nécessite 6.85. Cette différence d'un dixième signifie que la prédiction d'une particule sur 10 seulement doit être réajustée au cours de la déformation.

6.2. Collision entre particules

Le mécanisme de prédiction a été facilement amélioré pour gérer l'auto-collision entre les particules. Chaque volume possède un identifiant, nous permettant ainsi de créer une liste de particules contenues dans chacun des volumes. Dans cette liste, on vérifie la présence ou l'absence de collision entre chacune des particules avec une complexité en $O(n^2)$ avec n le nombre de particules présente dans la cellule. Sans avoir implanté de simulation physique réaliste, il nous a été permis de simuler de manière interac-

tive 1000 particules sous l'effet de la gravité avec tests d'auto-collision dans le maillage présenté sur la figure 14. Le nombre d'images par seconde varie de 100 à 300Hz selon la concentration de particules dans une cellule.

6.3. Comparaison avec une méthode hiérarchique

La comparaison suivante a été mise en oeuvre sur un PC de 2.4GHz Inter Core2 avec 2Go de mémoire. Seul un core a été utilisé pour la simulation.

6.3.1. Expérimentation

Nous comparons ici notre algorithme avec une hiérarchie de boites englobantes de type AABB. L'implantation de cette hiérarchie provient de la librairie physique Bullet ; librairie utilisée pour la simulation physique de Blender. Nous avons simulé le comportement de 1000 particules pendant 1000 pas de temps avec des complexité croissante. Les particules sont initialisées à une position aléatoire et subissent la gravité. L'environnement est formé de n cubes, nous modifions ce paramètre afin d'augmenter la complexité, tel que montré sur la figure 16.

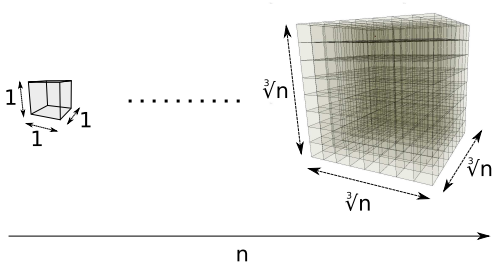

Figure 16: *Construction d'une scène de taille n pour la comparaison.*

La hiérarchie de AABB est utilisée pour limiter le nombre de tests d'intersection segment/triangles dans la scène. Les segments sont définis selon le point de départ et le point d'arrivé d'une particule pour un pas de temps donné. Le temps de calcul de la physique est ignoré dans les deux simulations.

6.3.2. Analyse des résultats

Le graphique 17 montre les résultats numérique de l'expérience. Notre méthode utilise moins de temps pour les tests de détection de collision. Cette différence s'explique par le fait que la hiérarchie de AABB n'utilise pas la cohérence temporelle. De manière encore plus significative, on peut observer la forme des deux courbes. La complexité de notre méthode reste constante alors que la complexité de la scène augmente. Avec les même paramètres, la hiérarchie de AABB voit son temps de calcul augmenter en $log(n)$.

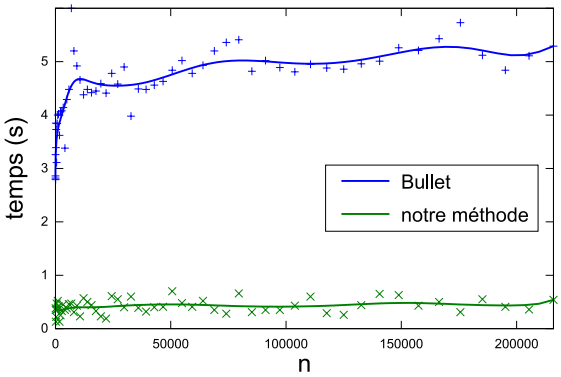

Figure 17: *Temps de calcul passé pour la détection de collisions en fonction de la complexité de la scène pour 1000 particules durant 1000 pas de temps.*

7. Discussion

7.1. Changement de topologie

Grâce à la structure topologique sous-jacentes, des modifications de la subdivision sont aisément envisageables. Bien qu'elles n'ont pas été implantées, nous donnons ici les méthodes à mettre en oeuvre pour permettre ces modifications.

Un changement de topologie peut se résumer en trois opérations dans notre cas : la couture, la fusion ou le découpage de cellules. Notre algorithme se basant sur la localité des tests, seules les particules se trouvant dans les cellules modifiées sont potentiellement à mettre à jour.

La couture consiste à relier deux faces qui étaient séparées afin de créer une liaison entre elle. Ce cas ne pose aucun problème, toutes les prédictions s'effectuant de manière locale aux cellules, le nouveau lien est utilisé automatiquement au pas de temps suivant.

La fusion consiste à supprimer une $(n-1)$-cellule entre deux n-cellules. Deux cas se produisent :

- aucun des brins de la $(n-1)$-cellule n'est utilisé pour une prédiction, dans ce cas aucun changement n'est nécessaire ;
- au moins une particule vise un des brins de la $(n-1)$-cellule. Dans ce cas il suffit de prendre arbitrairement une autre $(n-1)$-cellule de la n-cellule qui n'est pas affectée par la fusion. On relance ensuite une phase de détection de collision afin de corriger le triangle/tétraèdre de prédiction.

Le découpage consiste à ajouter une $(n-1)$-cellule entre deux n-cellules. Trois cas peuvent alors se produire :

- p_t et p_{t+1} se trouvent de chaque coté de la $(n-1)$-cellule ajoutée, on prend alors un brin de celle-ci. En relançant l'algorithme sur un pas la prédiction est alors corrigée ;

- p_t et p_{t+1} se trouvent du même coté de la $(n-1)$-cellule et la prédiction est encore correcte ;
- p_t et p_{t+1} se trouvent du même coté de la $(n-1)$-cellule mais le brin visé se trouve dans la cellule adjacente (de l'autre coté de la $(n-1)$-cellule). On effectue alors un nouveau pas d'orientation en prenant un brin de la $(n-1)$-cellule.

Lorsque le maillage est déformé de manière conséquente, il peut être nécessaire d'effectuer des modifications topologiques afin de respecter l'intégrité de la carte combinatoire sous-jacente. La méthode 2D décrite dans [LSM07] peut être utilisée comme base d'implantation dans ce but.

7.2. Subdivision hiérarchique

Lors de tests de collisions entre particules, il peut arriver que beaucoup de particules se trouvent dans une même cellule, le mécanisme est alors fortement ralenti. Ce cas ne devrait vraisemblablement pas arriver dans une application gérant un objet plus ou moins rigide à l'intérieur du maillage. Dans le cas où cela devrait éventuellement arriver, un mécanisme de subdivision hiérarchique permettrait de résoudre ce problème en découpant localement les faces ou volumes contenant trop de particules. De la même manière, si peu de particules se trouvent dans une même cellules, il est envisageable de les fusionner.

Lors des subdivisions ou fusions de cellules, les prédictions doivent être mises à jour, tel qu'indiqué dans la section 7.1, avant de pouvoir passer à la prédiction du pas de temps suivant.

Une analyse plus poussée doit être effectuée afin de définir de manière claire les configurations pour lesquels le temps passé à effectuer une subdivision ou une fusion de cellules est compensé par le temps gagné pour les tests de collisions entre particules.

7.3. Précision numérique

Lorsque les calculs sont effectués avec des nombres à précision finie la question de la stabilité numérique peut se poser. Les passages d'un état à un autre sont définis de telle manière que l'état le plus contraignant est conservé. En effet, lorsque les valeurs numériques de tests d'appartenance à une *i*-cellule, une arête ou un sommet par exemple, ne sont pas consistantes pour toutes les entités la composant, la contrainte la plus forte est retenue. Par exemple, si l'intervalle défini par deux plans, dans le cas d'une arête, présente une ambiguïté au niveau numérique, on considérera que la particule se trouve toujours sur l'arête. Le problème des erreurs numériques sur les tests d'orientation est traité de manière approfondie dans [She97].

De manière générale, le passage par une arête ou un sommet est rare dans une simulation. Il est possible de définir une marge dans les intervalles permettant l'apparition plus fréquente de ces états. Cette marge peut éventuellement être utilisée dans un cadre de modélisation géométrique avec un système à retour d'effort. Ce type de système peut alors guider l'utilisateur le long d'une arête pour l'aider à la sélection d'entités géométriques.

8. Conclusion

Cet article présente un mécanisme de prédiction pour la détection de collisions. Un système de prédiction est défini pour un point se déplaçant dans un environnement subdivisé. Nous traitons chaque cas particulier afin d'éviter l'apparition de cas dégénérés. Le système de prédiction est adapté à la navigation d'une arête à l'intérieur de la subdivision, permettant ainsi l'intégration à des applications telles que les simulations d'endoscopies.

La structure topologique nous permet de définir clairement et de manière robuste le système de prédiction. Cette dernière ayant d'ailleurs l'avantage d'être adaptée à tout type de subdivisions en polyèdres quelconques convexes.

L'utilisation des cartes combinatoires peut de plus permettre le couplage à de la multirésolution [KCB08] pour permettre d'adapter la précision de la simulation durant la navigation. Cette propriété est intéressante si le principe de synchronicité – une vraie seconde doit être égale à une seconde simulée – est une contrainte de l'application.

L'algorithme est opérationnel et permet d'ores et déjà de manipuler aisément plusieurs milliers de particules de manière interactive avec gestion de l'auto-collision ou encore avec déformation du maillage.

Finalement, notre système de prédiction est limité à la simulation de particules. Nous prévoyons d'étendre notre méthode aux arêtes et aux faces afin de permettre la simulation de solides selon le même principe.

Références

[Aco04] ACOSTA O. : *De la navigation exploratoire virtuelle à la planification d'interventions endovasculaires.* PhD thesis, LTSI. University of Rennes 1., 2004.

[Bau75] BAUMGART B. : A polyhedron representation for computer vision. In *Proc. of AFIPS* (1975), vol. 44 de *National Computer Conference*, pp. 589–596.

[BGTG03] BIELSER D., GLARDON P., TESCHNER M., GROSS M. : A state machine for real-time cutting of tetrahedral meshes. *11th Pacific Conference on Computer Graphics and Applications* (2003), 377.

[Bri93] BRISSON E. : Representing geometry structures in *d* dimensions : Topology and order. *Discrete & Computational Geometry* (1993), 387–426.

[CD99] CAZIER D., DUFOURD J. : A formal specification of geometric refinements. *Visual Computer. Vol. 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 *SI3D '95 : Proceedings of the 1995 symposium on Interactive 3D graphics* (New York, NY, USA, 1995), ACM Press, pp. 189–ff.

[EG86] EDELSBRUNNER H., GUIBAS L. : Topologically sweeping an arrangement. In *Proc. of the 18th ACM Symposium on the Theory of Computing (Berkeley)* (New-York, May 1986), ACM, (Ed.), pp. 389–403.

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

[Gei00] GEIGER B. : Real-time collision detection and response for complex environments. In *CGI '00 : Proceedings of the International Conference on Computer Graphics* (Washington, DC, USA, 2000), IEEE Computer Society, p. 105.

[GHPT89] GANGNET M., HERVÉ J.-C., PUDET T., THONG J.-M. V. : Incremental computation of planar maps. In *Proc. of ACM-SIGGRAPH Conf. on Computer Graphics* (Boston, Massachusetts, 1989), vol. 23, pp. 345–354.

[GJK88] GILBERT E., JOHNSON D., KEERTHI S. : A fast procedure for computing the distance between complex objects in three-dimensional space. *Robotics and Automation, IEEE Journal of [see also IEEE Transactions on Robotics and Automation]. Vol. 4*, Num. 2 (Apr 1988), 193–203.

[GLM96] GOTTSCHALK S., LIN M. C., MANOCHA D. : Obbtree : a hierarchical structure for rapid interference detection. In *SIGGRAPH '96 : Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, USA, 1996), ACM Press, pp. 171–180.

[GS85] GUIBAS L. J., STOLFI J. : Primitives for the manipulations of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics. Vol. 4*, Num. 2 (avril 1985), 74–123.

[HKM95] HELD M., KLOSOWSKI J. T., MITCHELL J. S. B. : Evaluation of collision detection methods for virtual reality fly-throughs. In *In Canadian Conference on Computational Geometry* (1995), pp. 205–210.

[Hub95] HUBBARD P. M. : Collision detection for interactive graphics applications. *IEEE Transactions on Visualization and Computer Graphics. Vol. 1*, Num. 3 (1995), 218–230.

[Jou06] JOUSSEMET L. : *Approche évolutionniste pour la détection des collisions au sein d'environnements virtuels denses*. PhD thesis, Université Montpellier II, 2006.

[JTT01] JIMÉNEZ P., THOMAS F., TORRAS C. : 3D Collision Detection : A Survey. *Computers and Graphics. Vol. volume 25*, Num. 2 (2001), pages 269–285.

[KCB08] KRAEMER P., CAZIER D., BECHMANN D. : Extension of half-edges for the representation of multiresolution subdivision surfaces. *The Visual Computer. Vol. (to be published)* (2008).

[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 Transactions on Visualization and Computer Graphics. Vol. volume 4*, Num. 1 (1998), pages 21–36.

[LC91] LIN M., CANNY J. : A fast algorithm for incremental distance calculation. *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on* (Apr 1991), 1008–1014 vol.2.

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

[Lie91] LIENHARDT P. : Topological models for boundary representation : a comparison with n-dimensional generalized maps. *Comput. Aided Des.. Vol. 23*, Num. 1 (1991), 59–82.

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

[LSM07] LÉON P.-F., SKAPIN X., MESEURE P. : Animation événementielle de structures topologiques : application à la construction de chenaux. In *AFIG* (November 2007).

[LT07] LINDBLAD A., TURKIYYAH G. : A physically-based framework for real-time haptic cutting and interaction with 3d continuum models. In *SPM '07 : Proceedings of the 2007 ACM symposium on Solid and physical modeling* (New York, USA, 2007), ACM, pp. 421–429.

[OL03] OTADUY M. A., LIN M. C. : Clods : dual hierarchies for multiresolution collision detection. In *SGP '03 : Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (Aire-la-Ville, Switzerland, 2003), Eurographics Association, pp. 94–101.

[PB07] PEROCHE B., BECHMANN D. : *Informatique Graphique et rendu*. Traité IC2 - Information - Commande - Communication. Hermès, march 2007.

[She97] SHEWCHUK J. R. : Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete & Computational Geometry. Vol. 18*, Num. 3 (octobre 1997), 305–363.

[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 (EG-STAR)* (2004), Eurographics Association, pp. 119–139.

[Tut84] TUTTE W. : Graph theory. In *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1984, ch. 21.

[WDS*07] WANG F., DURATTI L., SAMUR E., SPAELTER U., BLEULER H. : A Computer-Based Real-Time Simulation of Interventional Radiology. In *27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE-EMBS)* (2007).