Action Spécifique CNRS N° 90
27 janvier 2005

Détection des collisions et Calcul de la réponse

Porteurs :

P. Meseure IRCOM/SIC – Université de Poitiers
A. Kheddar LSC – Université d'Evry–Val d'Essonne
F. Faure GRAVIR – INPG/Université Joseph Fourier, Grenoble

Adresse Internet : http ://www-evasion.imag.fr/Membres/Francois.Faure/ascollisions

Remerciements

Nous tenons à remercier les personnes suivantes pour leurs contributions à la rédaction de ce document :

- O. Galizzi, GRAVIR – INPG/Université Joseph Fourier, Grenoble
- S. Redon, GAMMA – Université Caroline du Nord, USA.

Nous remercions également l'ensemble des membres de l'AS qui ont contribué au recensement et à la description des recherches menées en France :

- G. Dumont, IRISA – Rennes
- O. Nocent, LERI – Reims
- A. Crosnier, LIRMM – Montpellier
- D. Chablat, IRCCyN – Nantes
- D. Cazier, LSIIT – Strasbourg
- J.P. Jessel, IRIT – Toulouse
- S. Coquillart, I3D – Grenoble
- M. Daniel, LSIS – Marseille
- S. Foufou, L2EI – Dijon
- E. Guilbert, IReNav – Brest
- C. Duriez, CEA – Fontenay–aux–Roses
- A. Savary, Simteam – Paris

Table des matières

Première partie

Bilan Scientifique de l'AS

Chapitre 1

Introduction

Ce document constitue le rapport final de l'action spécifique CNRS n° 90 intitulée « détection de collisions et calcul de la réponse » étude menée de novembre 2002 à décembre 2003 dans le cadre du Réseau Thématique Pluridisciplinaire n° 7 : *réalité virtuelle, synthèse d'images et visualisation.*

Ce rapport comporte deux parties. La première est un bilan scientifique concernant le déroulement de l'AS. Cette partie est dédiée à une lecture rapide permettant à qui s'intéresserait à ce problème de trouver les informations concernant le déroulement de cette AS, les acteurs de la recherche en France, en Europe et au niveau international. Cette partie introduit de manière globale la problématique de la détection de collision ainsi que les tendances en recherche. Ce que nous pensons être les perspectives de recherche et les problèmes de recherche à surmonter dans le domaine sont aussi présentés.

La deuxième partie de ce document est une annexe technique. Elle s'adresse aux spécialistes ou aux chercheurs soucieux d'avoir une idée approfondie de l'état de l'art en détection de collision et réponse. Plusieurs méthodes y sont décrites. Cette annexe comporte aussi une bibliographie fournie.

1.1 Problématique

La problématique de cette action spécifique est la *détection de collisions (DdC) entre objets virtuels* c'est-à-dire l'ensemble des méthodes permettant d'empêcher que deux objets ne s'interpénètrent ou de limiter leur interpénétration dans un environnement virtuel animé. Ce problème est étudié dans de nombreuses disciplines : géométrie algorithmique, robotique, synthèse d'images et réalité virtuelle. Car il se pose dans de nombreuses applications : la planification de trajectoires, l'animation assistée par ordinateur, la simulation basée sur la physique, les interactions entre objets en réalité virtuelle, la visualisation scientifique, le pilotage de dispositifs à retour d'effort, les jeux vidéo... De plus, lorsqu'une collision est détectée, il faut pouvoir alors en tirer les informations qui vont permettre de réagir à cette collision : c'est le *calcul de la réponse*, qui dépend étroitement de l'application.

Or, parce que de nombreuses disciplines se sont intéressées au problème, les approches sont particulièrement abondantes et il est difficile de sélectionner celles qui sont les mieux adaptées à un problème donné. Les états de l'art n'abordent assez souvent qu'une partie des solutions. Aussi, l'un des objectifs principaux de cette AS est d'établir un état de l'art aussi synthétique que possible.

La détection des collisions est un problème délicat dans les environnements virtuels : sa

complexité algorithmique est quadratique (en nombre d'objets ou parties d'objets) d'où des temps de calculs souvent trop importants pour les applications concernées. En outre, la robustesse de la résolution est souvent cruciale : toutes les collisions doivent être à la fois précisément détectées, et caractérisées.

1.2 Objectifs

Le but de cette AS est de recenser et de synthétiser l'état de l'art dans le domaine de la recherche en termes d'algorithmique de détection de collisions. Il s'agit notamment :

- d'identifier les butées théoriques et les problèmes d'implémentation pratiques ;
- d'identifier les besoins (éventuellement industriels) en termes d' algorithmique de détection de collision et réponse ;
- de mener une réflexion et faire des propositions sur les voies de recherche prometteuses autour des besoins (dictées essentiellement par l'applicatif) ;
- d'identifier les acteurs de la communauté française et internationale dont les recherches traitent de ce problème ;
- de recenser et évaluer les algorithmes ou/et les codes existants ;
- de réfléchir à une standardisation.

1.3 Participants

Les auteurs de ce document et porteurs de l'AS sont :

- Philippe Meseure, IRCOM/SIC, Université de Poitiers (meseure@lifl.fr)
- Abderrahmane Kheddar, LSC, Université d'Évry (kheddar@iup.univ-evry.fr)
- François Faure, GRAVIR/Univ. Joseph Fourier, Grenoble (Francois.Faure@imag.fr)

Les laboratoires suivants ont fait partie de l'AS :

- LIFL – Lille
- LSC – Évry
- GRAVIR – Grenoble
- IRCCYN – Nantes
- LIRMM – Montpellier
- IRISA – Rennes
- LSIIT – Strasbourg
- LERI – Reims

De nombreux laboratoires se sont joints aux membres initiaux :

- I3D – Grenoble (anciennement Rocquencourt)
- IRIT – Toulouse
- LE2I – Dijon
- ENSI – Bourges
- LSIS – Marseille
- IRENav – Brest

Enfin, des entreprises privées ou assimilées font également partie de cette AS :

- SIMTEAM – Paris
- CEA – Fontenay-aux-Roses

1.4 Déroulement de l'AS

Pour atteindre les objectifs de l'AS détection de collision, nous avons suivi le calendrier suivant :

- Décembre 2002 : Appel à participation
- 2 février 2003 : Réunion préparatoire à Grenoble, présentation de quelques travaux
- 22 mars 2003 : Présentation de l'AS au groupe de travail Modélisation Géométrique
- 2 mai 2003 : Réunion intermédiaire des membres de l'AS
- 17 juin 2003 : Organisation d'un séminaire sur la détection de collisions dans le cadre des journées du groupe de travail *Animation et Simulation*.
  Financement de la présentation invitée de S. Redon en post-doc dans l'équipe Gamma de l'Université de Caroline du Nord
- 8-9 octobre 2003 : Participation à la Session « Réalité Virtuelles et Interface »des 4èmes journées Nationales de Recherche en Robotique à Murol.
- 29 octobre 2003 : Première réunion d'élaboration du rapport final
- 3-5 décembre 2003 : Financement d'une présentation invitée aux journées de l'AFIG. Présentation de David Kirk (société nVidia) « The Future of Graphics Computing »(présentation des accélérations proposées par les cartes graphiques, certaines techniques étant employées pour la DdC matérielle)
  Participation à la session « Détection des collisions »

Les deux journées du 2 février et du 17 juin ont été consacrées à des exposés scientifiques, états de l'art ou résultats de recherche récents. Ces exposés ont grandement aidé la rédaction des états de l'art de l'annexe technique.

Les diverses actions menées et travaux présentés ont été regroupés à l'URL suivante : http ://www-evasion.imag.fr/Membres/Francois.Faure/ascollisions.

Une liste de diffusion a été créée : as-collis@univ-lille1.fr (voir http ://wwsympa.univ-lille1.fr/wws/admin/as-collis pour les archives et la liste des membres).

1.5 Plan du document

Ce document est construit de la façon suivante. Une première partie dresse le bilan scientifique de cette AS dont cette introduction forme le premier chapitre. Le deuxième chapitre expose les concepts fondamentaux de la détection des collisions. Le chapitre suivant présente les problèmes identifiés et les pistes de recherche qui en découlent. Nous dressons alors dans le chapitre suivant un tableau des divers travaux au niveau international, européen, puis français, pour terminer sur les tendances et recherches études menées actuellement par les membres de l'AS. Nous concluons ensuite cette première partie par les résultats obtenus et un bilan financier est dressé. La seconde partie du document est constituée par un état de l'art rédigé à l'occasion de cette AS.

Chapitre 2

Présentation de la détection des collisions

2.1 Positionnement du sujet

Plusieurs applications de la vie courante nécessitent des modules capables de prévenir des « collisions » entre entités de nature physiques diverses. En trafic aérien, maritime, ferroviaire voir même routier, il est nécessaire de prévenir, pour des raisons évidentes, les éventuelles collisions. Dans ces cas réels, les collisions sont prévenues par des capteurs divers communiquant, soit avec un module central, soit entre eux. La simulation informatique de plusieurs processus nécessite souvent la mise en œuvre d'un module spécial dont le but est d'identifier les instants et/ou les localisations de contacts ou d'interactions entres les entités simulées. Dans ce cas d'applications, on entend par *détection de collision* un capteur informatique (ou observateur) de contacts ou d'interactions. De nos jours, la simulation concerne un nombre très important d'applications. Ainsi, le type de *détection de collision* dépend beaucoup de ce que l'on souhaite simuler. Cependant, avant d'aborder l'algorithmique de la *détection de collision*, nous présentons quelques applications qui en font usage.

Dans le domaine de la robotique, la détection de collision se base essentiellement sur des capteurs intéroceptifs (internes au robot) et extéroceptifs (localisés au sein de l'environnement du robot). Ces capteurs permettent au robot d'interagir efficacement avec son environnement pour la réalisations de tâches diverses : navigation, montage/démontage... Ainsi, lorsqu'il s'agit de tâches de navigation, d'exploration ou de transport, le robot requiert des capteurs de proximité afin de lui permettre d'éviter des obstacles grâce au module de planification et de génération de trajectoires sans collision avec éventuellement optimisation énergétique. Ces capteurs sont souvent des senseurs de distance à base d'ultrasons, de balayage laser, de caméras... Pour des tâches robotiques de manutention ou de « manipulation » des senseurs tels que les capteurs de forces, tactiles ou tout simplement des contacteurs sont utilisés. Ces derniers détectent et renseignent souvent sur la nature des « contacts » effectués entre le robot et son environnement. Dans le cas des humanoïdes, un module de détection de collision permet la génération d'allures et l'évitement d'auto-collisions. La détection de collision est utilisée aussi dans la conception et le prototypage de lois de commande bas niveau référencées capteurs, elle s'intègre donc dans des « boucles réflexes ». En téléopération, l'algorithmique de détection de collision est surtout utilisée en téléprogrammation et programmation graphique et comme moyen d'assistance à l'opérateur (téléopération assistée par ordinateur).

Dans le domaine des transports, le GPS, le sonar et autres capteurs de positionnement sont

utilisés pour localiser les engins de transports divers (véhicules, avions, trains, etc.). Lorsqu'un même espace est partagé par plusieurs de ces engins de transport, il est souvent utile d'ordonner leurs circulation afin d'éviter les collisions. Dans le domaine spatial (évitement d'astéroïdes par des satellites) de l'aviation (autour d'un aéroport) ou de la marine (autour d'un port), les tours de contrôle sont équipés de tels systèmes. Dans ces cas, il s'agit de « traquer » les distances séparant les engins proches, ou autour d'une zone définie, pour prédire les collisions indésirables.

Enfin, les êtres vivants, l'homme et bon nombre d'animaux et de végétaux, disposent aussi de capteurs biologiques qui les renseignent sur les contacts (ou les distances) établis (ou qui les sépare) avec les objets de l'environnement. Pour l'homme, les capteurs de contact se trouvent dans la peau (mécanorécepteurs, poils), pour certains animaux ou végétaux se sont essentiellement les poils qui renseignent les contacts.

Le cadre de cette AS ne concerne que la détection de collisions utilisée en réalité virtuelle. Nous détaillons dans la suite les applications visées.

2.2 Les applications et leurs contraintes

Dans le cadre plus particulier des applications à base de techniques de réalité virtuelle, l'applicatif concerne, entre autres :

- les simulateurs interactifs ;
- les applications d'enseignement et d'apprentissage ;
- les cyberespaces ;
- les progiciels de CAO/FAO et prototypages ;
- la planification de trajectoire en robotique ;
- les divertissements ludiques, jeux, etc. ;
- l'art.

Dans toutes ces applications le terme *détection de collision* est souvent rencontré avec des dénominations diverses. On parle souvent de détection d'*intersections*, d'*interférences*, de *recouvrements* ou d'*interpénétrations* lorsqu'il s'agit de détecter si deux objets virtuels s'interpénètrent ou pas. On parle aussi de *détection de contacts* lorsqu'il s'agit de définir les localisations où les objets virtuels sont en contact, à cette appellation est associé souvent la notion sous-jacente de *topologie de contact* qui fait que l'on ne s'arrête pas à une « détection binaire ». Dans certaines applications, la détection de contact sous-entend aussi la possibilité de déterminer le premier instant de collision. Enfin, on parle aussi de *détection de proximité* lorsqu'il s'agit d'algorithmique liée à la détermination de seuils de distances ou la distance minimale séparant une paire d'objets virtuels.

Les contraintes globales dictées par les applications à base de technique de RV concernent essentiellement les aspects ;

- *interactivité* : la nécessité de satisfaire des contraintes temps réels liées aux processus (réels ou virtuels) qui peuvent être couplés à la simulation. C'est notamment le cas lorsqu'un opérateur humain interagit avec la simulation en manipulant un objet virtuel via des interfaces de stimulations sensorielles (retours visuel, auditif ou haptique). En effet, il est primordial de maintenir une cohérence sensorielle pour avoir une bonne « immersion » interactive ;
- *généricité* : dictés par le soucis de dégager des standards indépendants de l'applicatif, des modèles de représentations et des algorithmes de réponse ou de simulation dynamique ;
- *performance* : certaines applications de prototypage, notamment industrielles, ou de simulation de phénomènes physiques complexes peuvent exiger de la précision et de la robustesse dans la détection de collision.

Les besoins et les solutions, ainsi que des contraintes plus spécifiques, dépendent aussi du type de détection de collision et de l'application envisagés. Ainsi, lorsqu'il s'agit d'applications qui requièrent de l'évitement d'obstacle, les besoins algorithmiques concernent :

- la mise-à-jour et le suivi des distances critiques (les plus proches) séparant les objets ;
- la prédiction de l'évolution de ces distances ;
- la robustesse, seulement pour les applications réelles qui la requièrent ;
- des contraintes temporelles lorsque le temps de réactivité est crucial.

Pour les applications de simulations scientifiques de processus complexes (interaction moléculaire, physique nucléaire...) dès lors que le temps de réponse n'est plus problématique et que le résultat recherché requiert plus de précision, l'algorithme de détection de collision peut être conçu de manière simple (car la combinatoire n'est pas optimisée) et peut prendre en compte dans sa conception la précision fixée par l'opérateur. En général, le retour arrière (i.e. dans la simulation en subdivisant le pas discret) est autorisé pour affiner la détection.

Par contre, dans le cas des applications ludiques (jeux et divertissement à base d'images et d'animation de synthèse), les besoins s'expriment plutôt en simplicité mais sans soucis de précision ou de robustesse. Les développeurs de jeux souhaitent des packages d'algorithme de détection de collision prêts leur permettant : une créativité rapide de scénarios, de réutiliser facilement les codes, une intégration rapide et une utilisation facile. C'est la raison pour laquelle, des algorithmes de détection de collision entre formes géométriques simples ont été développés dans le cadre de ces applicatifs. Il s'agit essentiellement d'algorithmes de détection d'interférences entre volumes à géométrie usuelle (cubique, cylindrique, sphérique, conique, parallélépipédique...).

Pour les applications mettant en œuvre l'interfaçage haptique (cf. AS 131 du RTP 15) la détection de collision doit répondre à des contraintes d'interactivité et de robustesse importantes. Les aspects temps réel sont liés à la bande passante du rendu haptique qui est de l'ordre du KiloHertz pour la perception des forces de contacts, chocs et texture durant l'interaction entre entités virtuelles. Ces phénomènes haptique doivent être rendu à l'opérateur via des interfaces haptiques actives (i.e. motorisées) pour contraindre le mouvement de l'opérateur dans l'espace des contacts. La robustesse concerne le fait que l'opérateur est dans la boucle de la simulation qui doit maintenir une cohérence entre les rendus hatique, visuel et auditif. La détection de collision se doit aussi d'être robuste aux fluctuations des périodes d'échantillonnage et au comportement aléatoire de l'opérateur.

2.3 Principe

Toutes ces applications montrent ce que l'on entend par *détection de collision*. Elles montrent aussi que ce que l'on attend sur le plan général, et les contraintes à prendre en compte, dépendent de l'applicatif envisagé. Dans ce document on s'intéresse aux simulations en réalité virtuelle dont le schéma peut se distinguer selon les étapes suivantes :

1. acquisition des divers traqueurs et capteurs externes – forces externes...
2. résolution d'équations dynamiques pour l'animation
3. mise-à-jour provisoire de l'état de l'environnement virtuel
4. détection de collision
5. quantification des collisions détectées
6. réponse aux collisions
7. correction et mise-à-jour définitive de l'état de l'environnement virtuel

8. rendus (visuel, haptique, sonore...)
9. retour à l'étape 1, tant que la simulation n'est pas terminée.

Ce schéma montre une *tendance* d'étapes. En réalité certaines étapes sont confondues. Dans d'autres schémas, les étapes n'ont pas exactement cette séquence et cela dépend beaucoup de la conception du *solveur dynamique*.

À supposer que l'on ait pu résoudre le problème localement, i.e. entre deux paires de primitives composant un objet virtuel, une approche naïve de la détection de collision dans un environnement virtuel serait l'algorithme en $\mathcal{O}(n^2)$ suivant :

```
Pour i de 1 a nombre d'objets de la scene
Faire
    Pour j de i a nombre d'objets de la scene
    Faire
        Detection_de_Collision(i,j)
    Fait
Fait
```

On voit bien que la combinatoire est impressionnante, notamment si on prend en compte le fait que chaque objet (i) se compose de plusieurs primitives géométriques simples qui sont à tester avec celles de l'autre objet virtuel (j) ou avec celles de (i) si on teste les auto-collisions. Pour réduire cette complexité les algorithmes de détection de collision doivent procéder par étapes successives, après avoir éliminé les cas inutiles. Ainsi, les algorithmes de DdC peuvent se faire globalement en trois phases :

1. Phase accélératrice de recherche de proximité (*broad phase*) : c'est une étape de détection « grossière » permettant de trouver quels sont les couples d'objets susceptibles d'entrer en collision, parmi tous les couples possibles.
2. phase accélératrice de détection approximative (*narrow-phase*) : C'est une étape de détection permettant de déterminer les zones de collision potentielle.
3. phase précise ou *noyau* : dans cette étape, seules les « zones » de détection potentielles (résultats de la phase accélératrice) sont investiguées. C'est dans cette étape que l'on opère des détections entre paires de primitives bas niveau avec éventuellement, l'identification de la localisation et la connaissance de l'état exact du contact et de ses paramètres (quantification). Là encore des propriétés de cohérence peuvent être exploitées de la même manière que la phase accélératrice.

Cependant pour une animation dynamique, la simple détection de collision n'est pas suffisante et le « processus » de détection de collision doit alors s'effectuer en quatre étapes :

1. la phase accélératrice de recherche de proximité
2. la phase d'approche permet alors de suivre l'évolution des distances nécessaires aux approches par contrainte ou pour l'évitement d'obstacles
3. la détermination de contact permet de localiser les zones de collisions et les points de contact
4. l'analyse du contact qui permet de générer des groupes de contact pour optimiser le modèles dynamique et bien conditionner les matrices associées.

Les algorithmes de détection de collision sont à appeler régulièrement lors du mouvement des objets, mais peuvent prendre en compte le paramètre temps de façon diverses. Certaines approches purement *spatiales* ne tiennent pas compte de l'évolution temporelle des objets : elles cherchent uniquement les intersections entre objets suposés statiques. Cette recherche peut déboucher sur plusieurs résultats que l'on classe ici par ordre croissant d'information sur l'intersection :

- réponse booléenne ;
- distance de séparation entre objets, 0 ou négative si intersection ;
- détermination des surfaces en intersection ou localisation du recouvrement ;
- détermination du volume d'intersection ou d'une profondeur d'interpénétration ou d'un vecteur d'extraction ;
- calcul explicite de l'intersection.

Il incombe alors à la simulation de remédier seule au recouvrement des objets. À partir d'informations liées à la profondeur d'interpénétration il est possible de calculer des forces de pénalité qui vont faire tendre les objets vers la séparation. Si un vecteur d'extraction est connu, il est également possible d'imposer des nouvelles positions d'objets (non corrélées au mouvement de l'objet) pour qu'il ne se recouvrent plus.

Dans un certain nombre d'application, il est cependant nécessaire de connaître l'instant du contact. Dans ce cas, il faut employer des méthodes spatio-temporelles. Elles sont de deux types :

- les détections par interférence, qui exploitent les méthodes spatiales, dites de classe *discrète* ;
- les détections par interpolation, dites de classe *continue*.

Les techniques discrètes tentent de trouver l'instant de contact en échantillonnant le temps. Ceci est possible en appliquant un algorithme à réponse booléenne de façon dichotomique au cours du temps. Une autre solution consister à opérer un « retour arrière » (*back-tracking*) qui consiste, à partir des informations de collisions et du mouvement des objets, à déterminer leur position lorsque l'interpénétration a débuté. Enfin, à partir des informations de distance de séparation entre objet et leur vitesse, il est également possible d'évaluer approximativement à quel instant la collision doit être à nouveau inspectée : on parle alors d'*anticipation* de la collision.

Les techniques continues visent à déterminer précisément le temps du premier contact (à l'interpolation près). C'est particulièrement utile pour les rendus de chocs (approches d'animation à base d'impulsions) et cela permet l'intégration des contraintes de non pénétration dans les approches d'animation dynamique par contraintes. On peut également trouver des algorithmes de classe semi-continue, i.e. la détection se fait sur les espaces ou volumes balayés ou engendrés par le mouvement (on projette le volume de trajectoire 4D le long de l'axe du temps sur l'espace 3D).

Les algorithmes discrets détectent la collision sur la base d'une éventuelle interférence en chaque instant de simulation. L'hypothèse que formule cette classe d'algorithmes est la suivante : si il n'y a pas d'interférence entre les instants discrets $k-1$ et k (ou k et $k+1$) alors il n'y a pas de collisions (ou d'interférences) entre ces instants. En théorie, cette hypothèse peut s'avérer fausse dans plusieurs situations. C'est toutefois, de loin, la classe la plus utilisée en simulation pour sa rapidité, et le fait que l'on exploite des algorithmes d'optimisation et de calcul de distance performants. Les algorithmes continus ajoutent une dimension supplémentaire au problème : « le temps ». La trajectoire des objets virtuels est interpolée et paramétrée en temps entre deux pas discrets de la simulation. On cherche alors le premier instant de collision, ce qui donne aussi l'endroit et l'état « exact » de l'objet virtuel juste au moment du contact. L'hypothèse formulée

par cette classe d'algorithmes est la suivante : si entre deux pas discrets on détecte une collision, alors collision il y a. En théorie, ceci peut s'avérer faux seulement si la trajectoire issue de l'interpolation diffère de la trajectoire physique réelle[1] de l'objet virtuel.

Pour ce qui est des accélérateurs il existe essentiellement quatre approches dont certaines peuvent être mieux adaptées à un « noyau » discret et d'autres à un « noyau » continu :

- *les méthodes à base de subdivision spatiale* consistent (comme leur nom le laisse deviner) à subdiviser l'espace dans lequel évoluent les objets en régions et maintiennent une liste d'objets contenus dans chaque région. La détection de collision n'est alors effectuée qu'entre objets appartenant à la même région ;
- *les techniques topologiques* se basent sur un concept de bon sens : la DdC ne s'effectue qu'entre objets les plus proches. Ainsi une carte de proximité est construite puis maintenue au fur et à mesure que les objets se déplacent. Plusieurs techniques, issues de la géométrie algorithmique, existent pour la construction de la carte et sa mise-à-jour ;
- *les méthodes projectives* se basent sur une transformation simple : si des formes (ayant un certain nombre de propriétés) s'interpénètrent, alors toutes leurs projections (en général sur les axes du repère cartésien) s'interpénètrent. Pour séparer les objets, il suffit de trouver un axe tel que les projections soit disjointes. Cette axe est qualifié de *séparateur* (on parlera également de *plan séparateur* en considérant le plan orthogonal à cet axe) ;
- *les méthodes à base de volumes englobants* diminuent la complexité des algorithmes de DdC en recouvrant, à plusieurs niveaux, les objets par une hiérarchie de volumes englobants. Les volumes ont des formes simples qui facilitent d'une part leur construction et d'autre part la DdC entre eux. Plusieurs primitives de volumes ont été considérées : hiérarchie de sphères englobantes, de boîtes englobantes isothétiques ou alignées, les k–DOP (polytopes à orientation discrète).

Les méthodes continues et discrètes peuvent être combinées pour donner lieu à des accélérateurs hybrides : on peut combiner un noyau continu avec un accélérateur discret et vice-versa (ceci n'est pas conseillé car on pourrait avoir des problèmes d'incohérence).

Enfin nous proposons un ensemble de critères qualitatifs permettant d'évaluer les algorithmes développés dans le cadre d'applications à base de techniques de réalité virtuelle :

- *complétude* : la disponibilité de tout paramètre requis par la simulation ;
- *précision* : de la localisation, du temps de contact et de la zone de contact ;
- *robustesse et stabilité* : vis-à-vis de la variation du pas d'échantillonnage et de l'interactivité de la boucle de simulation ;
- *généricité* : vis-à-vis des modèles de représentation géométriques, des méthodes d'accélération de la DdC et du modèle d'animation dynamique ;
- *aboutissement à un standard* : la possibilité pour l'algorithme de pouvoir être implanter sous forme matérielle i.e. sur carte électronique ou carte graphique.

[1] En général non connue.

Chapitre 3

Propositions et perspectives de recherche

Nous exposons ici divers problèmes recensés par les membres de l'AS. Ce sont autant de perspectives de recherche pour lesquelles nous proposons éventuellement quelques pistes.

3.1 Détection de la collision

Nous détaillons dans ce paragraphe les problèmes que vous avons recensés concernant la détermination exacte de la collision, notamment les hypothèses sur le type d'objet géométrique utilisé et la détection employée.

3.1.1 Association Collision/Modèle d'objet

En fonction du type de modèle utilisé, il n'existe pas forcément de solutions permettant de calculer les diverses caractérisations de la collision que l'on peut désirer : distance inter-objet, calcul du volume, calcul du contour, mesure d'interpénétration...

Les polyèdres (en particulier convexes) sont les seules primitives présentant une panoplie d'algorithmes relativement complète. Les polyèdres ont en effet été les objets les plus étudiés, car de nombreux modèles peuvent être convertis en polyèdres. Au contraire, les objets définis par surfaces paramétriques n'ont que peu d'algorithme de détection de collisions à leur disposition. Ceci est d'autant plus paradoxal que les surfaces paramétriques de type splines et NURBS définissent la plupart des objets modélisés de nos jours. La plupart du temps, les utilisateurs utilisent une représentation polygonale. Le résultat n'est pas une « détection exacte » puisque la détection opère sur une approximation de l'objet. Pour plus de précision, il faut alors exploiter le résultat de la collision « polyèdrique » pour amorcer une recherche itérative plus précise. Cet enchaînement d'étapes rend la détection clairement insatisfaisante et surtout trop lente.

3.1.2 Hypothèse de convexité

Les algorithmes les plus performants calculent la collision entre polyèdres convexes uniquement. Lorsque cela n'est pas le cas, il faut accepter de travailler avec l'enveloppe convexe des objets ou traiter les objets comme une union de parties convexes. Il paraîtrait intéressant d'utiliser le polyèdre convexe comme primitive de base, comme le triangle l'est pour l'accélération graphique.

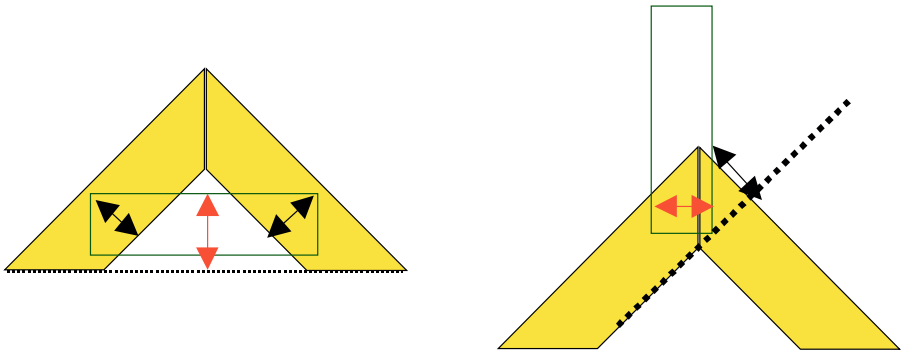

FIG. 3.1 – Problème de détection lors d'un découpage en parties convexes. À gauche, une décomposition en partie convexe fournit plusieurs profondeurs d'interpénétation, alors qu'une translation globale aurait peut-être été souhaitée (suivant l'application). À droite, la décomposition en parties convexes fournit également plusieurs profondeurs d'interpénétration (ici, de sens opposées) qui se révèlent inutilisables, la profondeur globale est ici la seule information exploitable.

Or cette décomposition est souvent insatisfaisante : le procédé de construction est lent (NP-dur), difficile à implanter[1], génère de nombreux éléments (faces, arêtes, sommets) superflus, et enfin revient à augmenter le nombre d'objets à prendre en compte [PML95]. En outre, la détection des collisions appliquée entre les parties convexes de deux objets concaves peut ne pas fournir le résultat attendu. Sur la figure 3.1, nous montrons qu'il est suivant les cas souhaitable d'obtenir une caractérisation unique de l'interpénétration (approche globale) ou bien une caractérisation de chaque partie connexe de l'intersection (approche locale), mais que ce résultat ne doit pas être dépendant de la décomposition en parties convexes choisie.

Enfin, il est légitime de se demander pourquoi les polyèdres convexes connaissent un tel intérêt. Il est souvent argumenté que les corps convexes ne présentent qu'un seul minimum local lorsque l'on cherche le point P (appelé *point support*) minimisant le produit scalaire $P \cdot \vec{V}$ pour une direction $\vec{V}$ donnée. Or, dans des cas dégénérés, il peut exister plusieurs points fournissant une même valeur minimale. C'est le cas d'un cube lors que $\vec{V}$ est l'une des six normales de ses faces. Ces cas dégénérés sont mal appréhendés par les algorithmes de détection qui fournissent des résultats incomplets ou instables dans le temps.

Compte tenu de tous ses inconvénients, il apparaît plus approprié de trouver des méthodes efficaces entre polyèdres quelconques plutôt que convexes.

3.1.3 Robustesse

Les algorithmes sont souvent peu robustes. Ils présentent la plupart du temps de nombreux cas dégénérés, nécessitant des traitements spécifiques. Pire, de nombreux algorithmes ne garantissent pas la convergence (ex : EPA, DEEP, SWIFT...). Une solution possible à ce problème de robustesse est l'utilisation de l'arithmétique d'intervalle, mais aucune approche n'a été proposée aujourd'hui.

Par exemple, Swift++ (un algorithme de détection de collision développé au groupe de recherche GAMMA de l'UNC [EL00]) traite les contacts selon une tolérance : en dessous d'une certaine distance inter-objet fixée par l'utilisateur, Swift++ considère qu'il y a contact. Le problème est qu'il ne renvoie que le contact dont la distance est la plus réduite (le meilleur contact

[1] À notre connaissance, il n'existe pas de bibliothèque domaine public réalisant une telle décomposition

selon lui). Or,cela est insuffisant dans le cas où plusieurs contacts existent et sont nécessaires (pour un modèle physique par contrainte, par exemple).

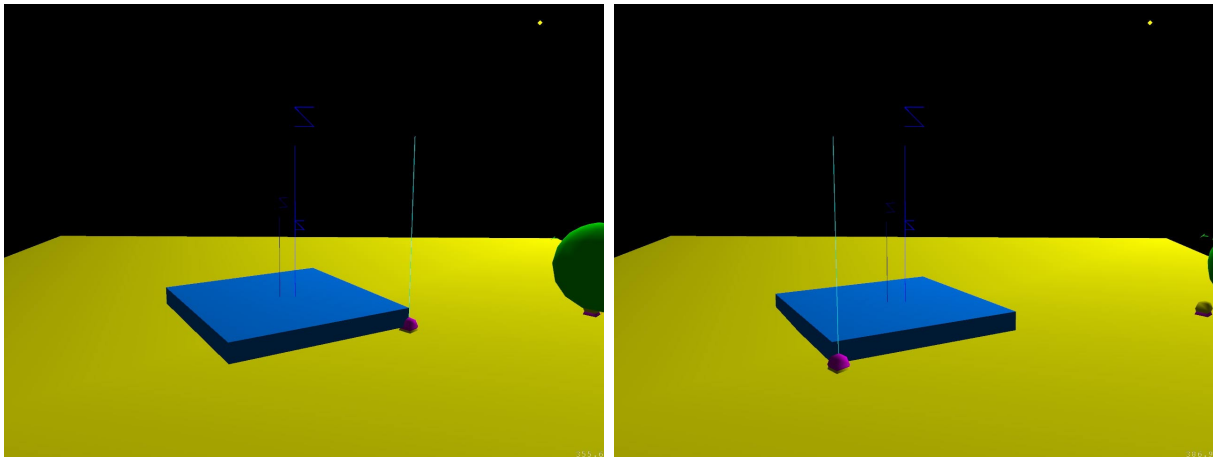

FIG. 3.2 – Le cube tombe sur le plan. Sur l'image de gauche, un seul contact est trouvé alors que les autres coins de la plaque sont négligés bien qu'ils soient à la même distance du support (le contact est symbolisé par la sphère se situant au coin de la plaque). Sur l'image de droite, au pas de simulation suivant, on voit bien que le point trouvé est différent du premier alors qu'il est à la même distance du plan que les autres. Là encore, un seul point est trouvé et ainsi de suite.

Par exemple, on peut considérer le cas d'un cube posé sur un autre cube : pour que le modèle physique accepte que le premier cube soit en équilibre sur le deuxième, il est nécessaire d'avoir au moins trois points de contact. Sinon, le modèle physique considère que le premier cube peut continuer à basculer. Il se crée alors une sorte d'oscillation entre les coins du cube qui deviennent alternativement plus proche selon son inclinaison (figure 3.2) : chaque coin du cube devient successivement un point de contact et la distance entre les deux cubes diminue.

La robustesse s'exprime aussi par la garantie d'un comportement unique en simulation hors-ligne (ou en ligne sans évènements extérieurs) ou interactive. Ainsi, dans le cadre d'applications haptiques, plusieurs algorithmes stables et efficaces développés dans le cadre d'animations graphiques montrent un comportement peu robuste voir instable en simulation interactive.

3.1.4 Détection continue

En pratique, la détection continue n'a été étudiée que dans le cas particulier des triangles. Même dans ce cas simple, les algorithmes (résolution d'équations de degré 2 ou 3, arithmétique d'intervalles) ont un coût non négligeable en temps de calcul car appliqué à de nombreux couples de primitives. Il s'agit donc d'étendre l'accélération à la détection continue : des techniques 2-body ont été proposées, mais peu de type N-body.

Par ailleurs, le mouvement intermédiaire utilisé pour la détection reste arbitraire (supposition de trajectoire rectiligne uniforme, hypothèse de mouvements rigides uniformes...). Dans une simulation dynamique, les schémas d'intégration des équations différentielles du mouvement se basent souvent sur des fonctions d'interpolation, prenant en compte non seulement les positions mais également les accélérations et vitesses. Il serait peut-être intéressant de combiner ces schémas d'intégration avec les routines de détection continue pour une détection plus précise.

3.1.5 Maîtrise du temps de calcul

Peu de méthodes de détection de collisions sont de complexité $\mathcal{O}(1)$. De ce fait, les temps de calcul ne sont pas bornés et se révèlent peu adaptées aux applications où la gestion du temps est cruciale (en animation interactive, pilotage de dispositifs à retour d'effort...). Pour ces applications, une utilisation d'architecture matérielle dédiée serait profitable (comme en synthèse d'images temps-réel). Certaines méthodes exploitant le matériel graphique sont proposées depuis quelques années, mais elles sont basées sur les primitives du rendu graphique (triangles) : ce sont donc des approches uniquement surfaciques qui ne fonctionnent pas lorsque la détection de collision doit fournir des informations de type volumique (profondeur d'interpénétration, volume d'intersection...).

3.1.6 Cas des corps déformables

Les algorithmes dédiés aux collisions existants font assez systématiquement l'hypothèse d'objets rigides. Les techniques de détection de proximité (voir chapitre 8) de type *sweep and prune* ou division de l'espace sont cependant facilement adaptables aux corps déformables. Au contraire, la plupart des méthodes de détection approximative exigent de bâtir des structures géométriques en pré-traitement. Lorsque la géométrie (et éventuellement la topologie) de l'objet se modifie, il faut adapter et le plus souvent reconstruire de telles structures : c'est le cas des décompositions en partie convexes et des hiérarchies de volumes englobants qui nécessitent des mises à jour à chaque déformation.

Ainsi, les méthodes compatibles avec les corps déformables se basent principalement sur des arborescences d'AABB [SKTK95, MHC98, Ber97] ou des k–DOP car leur mise à jour en cas de déformation est plus simple. La mise à jour complète de l'arbre a une complexité linéaire en fonction du nombre de primitives. Il faut alors ajouter la complexité de la détection proprement dite. La récente étude de [LM01] montre qu'une mise à jour par le sommet de la hiérarchie plutôt que par le bas est plus pratique, car elle réduit la mise à jour aux seuls volumes englobants utilisés. Cependant, cette mise à jour étant plus coûteuse (il faut parcourir tous les points contenus dans le volume), une méthode hybride est proposée : on ne met à jour qu'une portion supérieure de la hiérarchie après une déformation. Enfin, [LM01] indiquent qu'une construction exploitant les voisinages topologiques est généralement plus efficace après déformation qu'une construction exploitant les voisinages géométriques de l'objet non déformé [Ber97].

Cependant, toutes ces approches dont la hiérarchie est bâtie sur les primitive de l'objet, même si elle peuvent être adaptées aux corps déformables, perdent leur efficacité lors de changements topologiques de l'objet, comme les découpes ou les fractures. Dans ce cas, ce ne sont plus les simples volumes englobants de chaque nœud qu'il faut recalculer mais la structure même de l'arbre qu'il faut modifier. Pour toutes ces raisons, il semble que le découpage spatial, permettant une détection de complexité linéaire, ou les octrees sont plus intéressants [GDO00] que les méthodes à hiérarchie de volumes englobants pour les objets déformables et/ou à structure dynamique.

De façon générale, les méthodes de détection compatibles avec les corps déformables sont généralement au moins d'un ordre de grandeur plus lentes que les méthodes dédiées aux corps rigides. Actuellement, l'usage de corps déformables dans les simulation se fait avec parcimonie, ce qui est clairement insatisfaisant dans de nombreuses applications (ex : le médical).

3.1.7 Auto-collisions

La détection des auto-collisions peut exploiter les méthodes de détection des collisions standard, mais nécessitent de prendre en compte les liens de voisinages et la juxtaposition des parties de l'objet. De ce fait, nous constatons qu'il y a eu peu d'études. En outre, dans de nombreux cas, les auto-collisions peuvent être négligées (cas des objets élastiques peu déformables) et permettent ainsi un gain de temps appréciable.

On exploite souvent une décomposition de l'objet en petits éléments (triangles, tétraèdres...). La détection des auto-collisions est alors identique à la collision inter-objets [FL01]. Il reste cependant à détecter quels sont les éléments à inspecter. Pour les objets surfaciques (les tissus), de nombreuses méthodes ont été proposées. Certaines se basent sur la courbure [HDLM96, Pro97] ou sur des projections selon des directions données [VT94, VT95]. Une approche plus robuste a été proposée récemment puisqu'elle peut résoudre les auto-collisions dans une configuration de départ présentant déjà des interpénérations [BWK03]. Les accélérations sont cependant moins efficaces que pour deux objets distincts à cause de la proximité naturelle des primitives de l'objet entre elles. Des accélérations véritablement dédiées aux cas des auto-collisions restent encore à définir.

3.2 Calcul de la réponse

De nombreux problèmes pour calculer la réponse à une collision subsistent :

- Association de la détection et du calcul de la réponse : par exemple, peu d'algorithmes permettent de mesurer l'interpénétration. Au contraire, de nombreuses approches fournissent des triangles en intersection, données souvent insuffisantes pour le calcul de la réponse. On a alors recours à des algorithmes spécifiques pour remonter aux informations utiles [SL00] ;
- Le calcul de la réponse a souvent recours à des informations non géométriques comme la dynamique des corps ou l'historique du mouvement. Ces données sont parfois difficiles à extraire de l'algorithme de détection seul ;
- La plupart des algorithmes de collision se placent dans le cas de collision d'un point sur une surface. Cependant, les zones de collisions sont fréquemment définies à part de plusieurs points de contact, voire de surfaces de contact... ;
- Dans la réponse à la collision dans une simulation dynamique, il faut tenir compte des frottements entre objets. Ce calcul ne peut pas être traité a posteriori, car il conditionne globalement la réponse. Il faut alors évaluer les frottements et décider en cours de calcul si ce sont des frottements statiques ou dynamiques. Ce choix est délicat et les algorithmes peuvent parfois, lors de la prise de décision, osciller entre ces deux choix ;
- Les algorithmes de réponse ne savent pour la plupart traiter que les corps rigides. Il n'existe pratiquement aucune méthode permettant de déterminer précisément la réaction à la collision entre deux corps déformables, à savoir les forces et les déformations.

Ces problèmes concernent le calcul de la réponse entre deux objets. Cependant, lorsque plusieurs objets entrent en collision ou sont en contact en même temps (*multi-collisions*), il est souvent peu efficace d'appliquer les algorithmes de réponse traditionnels par couple d'objets en contact. Des méthodes de calcul de la réponse plus globales sont nécessaires [MS01].

3.3 Divers

Détection dans les environnements répartis

Les applications des environnements virtuels répartis se répandent rapidement (travail collaboratif, TP virtuels, jeux vidéo...). La détection des collisions est alors difficile à traiter, car les données stockées sur les machines distantes peuvent différer entre elles, ce qui rend le résultat de la détection aléatoire. L'utilisation d'un serveur dédié à la collision n'exploite pas le parallélisme inhérent aux architectures réparties et concentre les calculs. Cette solution n'étant pas satisfaisante, il s'agit d'adapter les algorithmes de détection en incluant des aspects « probabilistes » et utiliser des approches de type « *Dead Reckoning* »[LWTC99].

Détection pour les scènes volumineuses

Pour des scènes très volumineuses, il est généralement impossible de charger en mémoire l'intégralité de la scène. Or la détection de collision peut survenir entre deux objets quelconques de la scène. On utilise alors un cache permettant de garder certains objets en mémoire. On ne charge les objets utiles que s'ils ne sont pas déjà présents dans le cache. Cette étape ralentit considérablement la détection. Il s'agit de trouver des structures de données et un ordonnancement de la détection permettant de traiter conjointement la récupération des données sur disque et la forte probabilité de collisions : ainsi, un objet présent en mémoire est alors inspecté avec tous les objets avec lesquels il est susceptible d'entrer en collision.

Absence et difficulté pour définir des critères de comparaison Des études ont tenté de comparer certaines méthodes de détection. Cependant, les résultats sont souvent contradictoires, car ils dépendent fortement de l'application considérée et de la scène utilisée (grand ou petit nombre d'objets, complexité des formes, forte proximité ou non, vitesses, etc.). En outre, les études ne sont jamais complètes et de nombreux algorithmes pourtant essentiels ne font pas partie des expérimentations. Le temps de détection est certes un critère de comparaison important, mais il n'est pas le seul.

3.4 Bilan

Nous avons recensés divers problèmes concernant la robustesse des méthodes existantes et leur inadéquation à traiter certains types de modèles géométriques ou mécaniques, ou leur incomplétude au niveau des informations attendues.

Ces divers problèmes empêchent à l'heure actuel l'établissement d'un standard, comme il peut en exister un en affichage graphique. Établir les spécifications d'un pipeline de détection de collision souple semble cependant réaliste. Pour un maximum de souplesse, il faudrait pouvoir spécifier quel type de détection de proximité, de détection approximative (avec quel type de volume englobant) puis quelle détection exacte l'utilisateur souhaite utiliser.

Chapitre 4

Positionnement de la problématique

4.1 Niveau international

La détection des collisions est un sujet de recherche très étudié dans le monde dans des contextes aussi divers que la robotique, l'animation ou la géométrie algorithmique. Cependant, cette problématique est rarement un thème à part entière, mais plutôt une partie incontournable d'un sujet de recherche donné : simulation basée sur la physique, planification de trajectoire, etc. Ainsi, la détection de collisions est souvent englobée dans une autre problématique. C'est pourquoi ce sont moins des laboratoires mais plutôt des individus qui sont identifiés au niveau international.

Il existe cependant une exception majeure à ce constant. L'équipe Gamma de M.C. Lin et D. Manocha domine au niveau international le sujet de la détection de collision. Ce groupe de recherche très prolifique travaille à la conception et au développement d'algorithmes géométriques pour la modélisation et l'animation. Dans ce cadre, les thèmes de recherche incluent entre autres la détection de collision et les requêtes de proximité, la simulation du retour d'effort et ses applications, le parcours de modèles géométriques complexes, la planification de mouvement, la simulation fondée sur la physique (niveaux de détail en animation, simulation de chevelure, corps déformables, etc.), et l'interaction en temps réel avec des environnements virtuels. Une grande part des solutions de détection des collisions proposées ces dernières années sont issues de ce laboratoire : la plupart sont disponibles sous la forme de bibliothèques téléchargeables gratuitement. Leurs solutions sont donc utilisées par de nombreux laboratoires et industriels dans le monde. Cependant, ces bibliothèques traitent essentiellement du problème de la détection et peu de la réponse. Ainsi, leurs solutions sont souvent considérées comme rapides, mais incomplètes car elles ne fournissent pas toutes les informations dont les applications comme la planification de trajectoire ou la simulation physique ont besoin.

Bien sûr, d'autres chercheurs dans le monde ont consacré une partie de leurs études aux collisions. Par exemple, G. Baciu de l'université de Hong Kong a proposé diverses solutions basées sur les cartes graphiques. Johnson et Cohen, de l'université de l'Utah, ou P. Hubbard de l'université de Cornell ont apporté des contributions majeures à la détection de collision par volumes englobants hiérarchiques.

En géométrie algorithmique, les universités de Princeton (D. Dobkin et B. Chazelle), de Stanford (L. Guibas) et de British Columbia (D. Kirkpatrick) ont grandement contribué au domaine. En robotique, les recherches sur les distances inter polyèdres convexes sont issus des travaux de E. Gilbert de l'université du Michigan.

En simulation physique, les méthodes de calcul de réponse ont été et restent un sujet majeur d'études. D. Baraff et A. Witkin (Université de Carnegie-Mellon puis société Pixar studio) et

B. Mirtich (Université de Californie puis Mitshubishi Electronics Research Laboratories) ont une reconnaissance internationale sur ces sujets. Les laboratoires dans lesquels ces chercheurs travaillent actuellement montrent l'intérêt des industriels pour cette technologie (simulation, jeux vidéo, audio-visuel).

4.2 Niveau européen

Sur le plan européen, plusieurs personnes ont une reconnaissance mondiale dans le domaine. Ainsi, Stephen Cameron (Université d'Oxford) est l'un des premiers européens à avoir travaillé dans le domaine et proposé régulièrement des solutions, issues principalement des problèmes de robotique. Gabriel Zachmann (Université de Bonn), Carol O'Sullivan (Université de Dublin), Martin Held (Université de Salzburg) et Gino Van den Bergen (Université de Eindhoven) ont fourni des contributions importantes en détection hiérarchique par volumes englobants et ont chacun étudié un type particulier de volume.

Dans le domaine de l'animation, les équipes de l'EPFL et de MediaLab à Genève ont étudié certains problèmes de détection : des solutions à base de volumes englobants ou décomposition de l'espace ont été étudiées ainsi qu'une techniques d'auto-collision pour les tissus. En géométrie algorithmique, il faut mentionner les travaux de Elmar Shömer à L'institut Max Planck et ceux de Torras et Thomas à l'université de Barcelone.

4.3 Niveau national

Comme dans le monde, la détection des collisions est souvent en France un sujet de recherche complémentaire à une autre problématique principale. Elle est donc plutôt considérée comme une étape ou un outil nécessaire. Ainsi, les chercheurs ont utilisé des routines existantes ou ont proposé leur propre solution de détection. Cependant, ils reconnaissent souvent que les solutions utilisées ne répondent pas complètement à leur attente.

En France, les équipes pionnières de l'animation basée sur la simulation physique comme le projet Evasion (ex Imagis) de GRAVIR, l'ACROE à Grenoble ou le projet SIAMES de l'IRISA ont proposés dans les années 80 des méthodes de détection et de traitement des collisions. Depuis de nombreuses équipes ont fait de la simulation un de leurs sujets d'étude principaux : le projet EPIDAURE à Sophia-Antipolis, le LIFL à Lille, le LERI à Reims, le projet SHARP à Grenoble. Ces recherches ont elles aussi abouti à des méthodes de détection ou traitement des collisions. En particulier, l'équipe SHARP a proposé de nombreuses approches et reste très active dans ce domaine à la fois pour les corps rigides convexes et pour les corps déformables. Plus récemment, certaines équipes ont été confrontées au problème de la détection de collision pour les applications qu'elles visent (réalité virtuelle, applications médicales...). Par exemple le LSIIT de Strasbourg ou l'IRIT de Toulouse.

En robotique également, les recherches ont été actives, notamment en planification de trajectoires (LIRMM, IRCCyN, LAAS, LRP et bien d'autres).Les recherches en retour d'effort ont également amené des équipes à traiter de la détection de collisions : d'abord au LRP à Vélizy puis maintenant au LSC à Evry en collaboration avec le projet I3D de l'Inria, la détection des collisions est un sujet de recherche incontournable pour le retour d'effort, ainsi qu'au LIFL de Lille.

Enfin, sur un plan plus mathématique, les études menées en modélisation par le LSIS de Marseille ou le LE2I de Dijon font parties des travaux français sur la détection de collision appliquée à la CAO.

4.4 Les tendances actuelles

Les publications internationales de ces dernières années présentent des orientations très discernables. Les recherches se focalisent d'une part sur les systèmes de collisions à temps critiques ou bornés et d'autre part sur l'utilisation des cartes graphiques ou la conception de matériel spécifique. En calcul de réponse, ce sont les réponses incluant les frottements ou impliquant plusieurs objets en collision qui font l'objet des publications récentes majeures. Enfin, de plus en plus d'études traitent de la comparaison des méthodes existantes.

4.5 Les études menées par les membres de l'AS

• GRAVIR, Grenoble

L'équipe EVASION utilise la détection de collisions pour l'animation interactive de modèles physiques : blocs rigides (rochers), objets élastiques (tissus biologiques, cheveux, vêtements) voire fluides. Elle a effectué des travaux sur la collisions entre surfaces implicites. Elle s'intéresse particulièrement à des méthodes permettant de réaliser un compromis réglable entre précision et temps de calcul, et a développé dans ce cadre une méthode stochastique exploitant la cohérence temporelle sur un nombre réglable de paires d'éléments géométriques.

• LIFL, Lille

L'équipe Graphix du LIFL s'intéresse à la réalité virtuelle basée sur la physique. Des applications comme le prototypage en travail collaboratif synchrone ou la simulation médicale sont visées. Dans ce contexte, la détection des collisions est étudiée sur plusieurs plans : la détection et le calcul de la réponse entre objets de nature physique diverse (corps rigides ou déformables), collision pour les dispositifs à retour d'effort et détection dans les environnements répartis.

J. Dequidt, L. Grisoni, P. Meseure et C. Chaillou, « Détection de collisions entre objets rigides convexes autonomes », *Revue internationale de CFAO et d'informatique graphique*, Hermes Scienceq 2003.

J. Dequidt, L. Grisoni, P. Meseure et C. Chaillou, « Détection de collisions entre objets rigides convexes autonomes », *Actes des 15èmes journées de l'AFIG*, Lyon, 9-11 décembre 2002.

P. Meseure, J. Davanne, L. Hilde, J. Lenoir, L. France, F. Triquet et C. Chaillou, « A Physically-Based Virtual Environment dedicated to Surgical Simulation »*Surgery Simulation and Soft Tissue Modeling (IS4TM'03)*, Juan-les-pins, France, 12-13 juin 2003, pp 38-47.

J. Davanne, P. Meseure et C. Chaillou, « Stable Haptic Interaction in a Dynamic Virtual Environment »*Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2002*, Lausanne, 30 septembre-4 octobre 2002.

P. Meseure, L. Hilde et C. Chaillou, « Accélération de la détection de collisions entre corps rigides et déformables », *Actes des 6èmes journées du Groupe de Travail Réalité Virtuelle*, Paris, 12-13 mars 1998, pp 167-174.

• LSC, Évry

Dans le cadre de ses activités en « réalité virtuelle et haptique », le LSC développe des recherches en infohaptie (informatique du rendu haptique) dont la partie détection de collision prend une place importante. Les contraintes spécifiques du rendu haptique obligent à concevoir

des algorithmes temps réels, et robuste vis-à-vis du comportement aléatoire de l'opérateur. Le déterminisme de l'algorithme est une propriété importante pour la qualité de la commande (au sens de l'automatique). Les applications visées sont : prototypage virtuel en industrie, applications médicales, simulateurs interactifs de conduite, téléopération, téléprésence virtuelle.

• IRISA, Rennes

Dans le contexte de la réalité virtuelle, nous avons besoin de traitement rapide et réaliste de la collision ou du contact entre objets d'un environnement. Nous travaillons sur l'intégration d'algorithme de détection répondant au cahier des charges de la réalité virtuelle (i.e. temps réel du fait de la présence d'un opérateur dans la boucle). Le traitement du contact et de la collision nous apparaissent, du point de vue mécanique, comme deux traitement distincts. Nous avons développé un algorithme piloté par le mouvement et basé sur les configurations des objets dans la scène ainsi que sur leur vitesse qui discrimine les cas de contact et de collision. Le contact est alors traité par liaisons unilatérales au moyen de l'algorithme des contraintes actives dont la convergence est prouvée. Le choc est traité par pénalisation ou par coefficient de restitution, ce qui donne de bons résultats, en termes de stabilité et de rapidité d'exécution, sous réserve d'un bon calibrage des constantes du modèle. Nous avons également apporté un soin tout particulier au traitement d'interactions multiples. Les applications visées sont le montage (ou le démontage) pour les produits manufacturiers au sein, par exemple, du projet RNTL PERF-RV. Nous visons également le prototypage virtuel d'endoscopes actifs afin d'optimiser leur conception : dans ce cas, il faut un modèle de comportement des tissus humains ainsi qu'un modèle de restitution.

G. Dumont, « Algorithme des contraintes actives et contact unilatéral sans frottement », *Journal européen des éléments finis*, 4(1) :55–73, 1995.

F. Beauchamp, « Gestion des interactions en animation », thèse de l'Université de Rennes 1, decembre 1998.

T. Meyer, « Retour d'effort et réalité virtuelle : Proposition d'une boîte à outils pour l'intégration logicielle générique de la détection de collision et de la simulation physique », thèse de l'Université de Rennes 1, octobre 2003.

• LERI, Reims

L'équipe « Modélisation et Animation Dynamique pour la Simulation » (MADS) s'attache à proposer des modèles informatiques inspirés du formalisme lagrangien pour simuler des phénomènes physiques. Détachés des contraintes de temps-réel, les membres de l'équipe s'intéressent plutôt à la précision des modèles choisis. A ce titre, nous sommes à la recherche de méthodes géométriques de détection de points de contact, éventuellement multiples, de façon à pouvoir exprimer les réponses exactes nécessaires à la préservation du contact ou la gestion du choc. Ce calcul, conforme aux équations de la physique, devra prendre en compte les phénomènes de frottement et d'adhérence inhérents au contact.

• LIRMM, Montpellier

Le LIRMM mène des travaux sur la détection de collisions qui constitue une fonction importante en Robotique car elle est sous-jacente à plusieurs problématiques auxquelles il est nécessaire d'apporter des solutions afin de concevoir un système robotisé. Parmi ces problématiques, le LIRMM s'intéresse plus particulièrement aux aspects suivants :

– la génération et la planification de mouvements sécurisés ;
– la modélisation des interactions robot(outil)/environnement ;
– la mise en œuvre de techniques de réalité virtuelle (ou augmentée) pour l'aide au geste métier.

Des contraintes spécifiques au domaine sont de plus à prendre en compte : contrainte de temps réel, définition et complexité des environnements, caractéristiques physiques des environnements.

S. Druon, A. Crosnier, et L. Brigandat, « Efficient cellular automata for 2D/3D free-form modelling », *Journal of WSCG*, vol. 11, N° 1, ISNN 1213-6972, 2003, p. 102-108

A. Crosnier et S. Druon, « Human centered system for computer aided replication of sculptures », *Proc of the IROS 2003 Conference*, Las Vegas, p. 3751, 2003.

S. Druon et A. Crosnier, « Virtual sculpture : a model of virtual clay », *Proc. of the 9th Int. Conf. On Virtual Systems and Multimedia*, Montréal, Canada, p. 604-609, 2003

A. Lécuyer (IRISA, Rennes), C. Andriot (CEA/LIST) et A. Crosnier, « Interfaces haptiques et pseudo-haptiques », *Actes des 4ième Journées Nationales de Recherche en Robotique*, Clermont-Ferrand, France, p. 43-48, 2003.

• IRCCyN, Nantes

Les activités de l'IRCCyN en détection de collision sont liées à la manipulation d'objets, de robot ou de mannequins dans des environnements encombrés de types maquette numérique (DMU). Deux équipes de l'IRCCyN participent à l'AS détection de collisions. La première est l'équipe MCM (acronyme de Méthodes de Conception en Mécanique) et la seconde le projet IVGI (acronyme de Ingénierie Virtuelle pour le Génie Industriel). De nombreuses approches ont été utilisés dans ces équipes pour éviter les interfèrences entre les objets manipulés et leur environnement dans le cadre de la manipulation interactive ou les génération de trajectoires automatique ou assistée. Les contraintes et les informations demandées par chaque application étant différentes, il n'existe pas de réponse unique pour la détection de collision. Cependant, dans toutes nos applications l'absence d'interférence est une contrainte forte qui doit être vérifiée à chaque instant.

P. Chedmail, D. Chablat et C. Leroy, « A distributed Approach for Access and Visibility Task with a Manikin and a robot in a Virtual Reality Environment », *IEEE Transactions on Industrial Electronics*, 2003.

D. Chablat, F. Bennis, B. Hoessler et M. Guibert, « Périphériques haptiques et simulation d'objets, de robots et de mannequins dans un environnement de CAO- Robotique : eM-Virtual Desktop », *Mécanique et Industries*, Elsevier, 2003.

D. Chablat et F. Bennis, « Realistic Rendering of Kinetostatic Indices of Mechanism », *Virtual Concept*, Biarritz, 2003.

P. Chedmail, B. Maillé et E. Ramstein, « État de l'art sur l'accessibilité et l'étude de l'ergonomie en réalité virtuelle », *Mécanique & Industrie*, 2003.

B. Maillé, P. Chedmail et E. Ramstein, « Object and Manikin Path Planning - Multi-agent and Virtual Human Behaviour Emergence », *11 th World Congress in Mechanism and Machine Science*, 2004.

• LSIIT, Strasbourg

L'activité de l'équipe Informatique Géométrique et Graphique du LSIIT (Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection) se concentre autour de la synthèse d'images et des activités en programmation, spécifications et preuves qui lui sont liées. Un des thèmes de l'équipe est la spécification d'objets géométriques cellulaires en dimensions 2 et 3. Nous cherchons notamment à développer et à étudier différentes stratégies et algorithmes de navigation dans ces structures, interactive ou non, en suivant des critères topologiques et géométriques, et en prenant en compte la détection des collisions entre objets en déplacement. Ces études serviront de base aux applications que nous visons, notamment en imagerie médicale. Elles ont donné lieu à un premier prototype de navigation dans des objets triangulés du plan (stage de DEA de M. Daoudi en 2003, encadré par David Cazier et Jean-François Dufourd).

• IRIT, Toulouse

Dans l'équipe SIRV de l'IRIT, bien que ne travaillant pas directement sur tous les problèmes liés aux collisions, nous nous étions interessés à l'évitement de collision :

- En modélisation déclarative, pour le positionnement d'objets avec une gestion des contacts ou des positionnements complexes ;
- En animation et simulation comportemmentale : pour la construction de trajectoire sans collision pour tout ou partie d'un personnage.

Nous avons suivi les travaux de l'AS, nous sommes particulièrement intéressés par ses conclusions et prêts à poursuivre en travaillant plus directement sur :

- En modélisation déclarative : positionnement en équilibre, emboitements... ;
- Animation de personnages et Capture de mouvement : auto-intersection (peau, vêtements), contraintes pour l'édition de mouvement, ré-affectation ;
- Interaction haptique et Simulation : gestion des collisions (solides, objets déformables, fluides), choc, déformation, suivi de terrain, écoulement turbulent.

• I3D, Grenoble (anciennement Rocquencourt)

i3D est une équipe GRAVIR-INRIA Rhône-Alpes avec comme domaines de recherche principaux l'Interaction 3D et la Réalité Virtuelle. Récemment, le groupe i3D s'est en particulier intéressé à l'interaction haptique et a développé, en collaboration avec le LSC de lŠUniversité dŠEvry, un simulateur dynamique interactif dŠobjets rigides. Dans ce cadre, de nouvelles approches ont été proposées pour réaliser les deux composantes principales du simulateur : la détection de collisions entre les objets virtuels et le calcul de leur mouvement contraint.

La plupart des méthodes de détection de collisions sont discrètes : elles détectent seulement des interpénétrations entre les objets virtuels à des instants discrets successifs. Afin de détecter efficacement des collisions entre objets polyédriques rigides en continu, c'est-à-dire de calculer l'instant de premier contact entre eux et éviter les problèmes inhérents aux méthodes discrètes, il a été proposé d'utiliser un mouvement intermédiaire arbitraire pour remplacer le mouvement réel de l'objet et obtenir des équations de détection de collisions faciles à résoudre.

La plupart des méthodes classiques de calcul de mouvement contraint sont formulées dans l'espace des contacts. Il a été montré que, grâce au principe des moindres contraintes de Gauss, il est possible d'obtenir une formulation équivalente des problèmes dynamiques sans friction dans l'espace des mouvements, plus avantageuse sur le plan algorithmique.

Les algorithmes proposés ont été implantés et rassemblés dans une librairie C++, CONTACT Toolkit. Plusieurs applications utilisant cette librairie ont été développées. En interne pour des

simulations avec retour d'effort ainsi que pour des cas industriels fournis par Renault, PSA et Airbus-EADS.

S. Redon, « Fast Continuous Collision Detection and Handling for Desktop Virtual Prototyping » *To Appear in the Virtual Reality Journal* (Springer Verlag Ed.).

S. Redon, A. Kheddar et S. Coquillart, « Fast Continuous Collision Detection between Rigid Bodies » *Computer Graphics Forum, Eurographics*, 21 (3), 2002.

• LSIS, Marseille

Notre démarche scientifique dégage trois thèmes de recherche originaux :

– modélisation et contrôle des formes ;
– segmentation et reconnaissance des formes ;
– cohérence des modèles géométriques.

À l'intérieur de ces trois thèmes on retrouve le problème de l'intersection de surfaces : la modélisation d'objets complexes fait largement appel à des calculs d'intersection de surfaces qui doivent être rapides et fiables. Si la rapidité s'améliore, ne serait-ce que grâce à la puissance accrue des machines, la fiabilité reste un problème ouvert pour trois raisons : la précision, la détection et le traitement de tous les cas particuliers et/ou singuliers, et la détection de toutes les composantes de l'intersection. Nous nous intéressons dans ce contexte à élaborer des méthodes fiables et précises, et qui soient également rapides. Dans ce cadre, l'étude des auto-intersections de surfaces est un problème particulièrement intéressant.

M. Daniel, « A note on degenerate normal vectors », *Computer Aided Geometric Design*, 12, 1995, pp 857-860.

M. Daniel, « Using a Convex Pyramid to Bound Surface Normal Vectors », *Computer Graphics Forum*, volume 15 numéro 6, 1996, pp 219-227.

M. Daniel, « A Curve Intersection Algorithm with Processing of Singular Cases : Introduction of a Clipping Technique », dans *Mathematical Methods in Computer Aided Geometric Design II*, Tom Lyche and Larry Schumaker (eds), Academic Press, 1992, pp 161-170.

M. Daniel et A. Nicolas, « A surface-surface intersection algorithm with a fast clipping technique », dans *Curves and Surfaces II*, P.J. Laurent, A. Le Méhauté and L.L. Schumaker Eds., A.K. Peters, juillet 1994, pp 105- 112.

E. Malgras et M. Daniel, « Performance Improvement of a Surface-Surface Intersection Method », *IDMME'96 – Integrated Design and Manufacturing*, KLUWER Publ., P. Chedmail, J.C. Bocquet, D. Dornfeld eds., 1997, pp 475-484.

E. Kuzmin et M. Daniel, « Curves on Surfaces for Computer Graphics Applications : Theoretical Results », in *Curves and Surfaces with Applications in CAGD*, A. Le Mehauté, C. Rabut, L.L. Schumaker (eds.), Vanderbilt University Press 1997, pp 239-246.

E. Guilbert, E. Saux et M. Daniel, « A Hierarchical Structure for Locating Intersection in Large Sets of B-spline Curves », in *Curves and Surfaces Design*, Saint Malo 2002, T. Lyche, M.-L. Mazure, L.L. Schumaker (eds.), Nashboro Press, 2003, pp 205-214.

M. Daniel et A. Nicolas, « An hybrid surface-plane algorithm » *GRAPHICON'93, 3rd annual Conference for Computer Graphics and Visualization in Russia*, Saint Petersbourg, 13-17 septembre 1993.

E. Malgras et M. Daniel, « Control of Intersection Curves Computed with Marching Techniques », *Swiss Conference of CAD/CAM*, Neuchatel, 22-24 février 1999, pp 207-213.

B. Lacolle et M. Daniel « Intersections de courbes et surfaces paramétriques », *Séminaire courbes et surfaces Bézier B-splines (dans le cadre de l'ATP Mathématiques, Informatique et Applications du CNRS)*, Rennes 11-13 mai 1987.

• LE2I, Dijon

Les surfaces de subdivision constituent un moyen très attractif pour modéliser des formes arbitraires. Étant donné un modèle polygonal (ensemble de sommets, d'arêtes et de faces) constituant une description très grossière d'une forme quelconque, la subdivision de ce modèle consiste en l'application d'un ensemble de règles pour obtenir un modèle plus raffiné. La surface résultante après quelques étapes de subdivision (surface de subdivision) décrit la forme de façon beaucoup plus précise que le modèle initial. Au cours des dernières années, les surfaces de subdivision ont reçu beaucoup d'attention de la part des professionnels des images de synthèse que ce soit en recherche ou en industrie. Dans la production cinématographique et dans les jeux vidéo, les techniciens appliquent cette technique pour créer des acteurs de formes complexes et produire des animations de haut niveau (Geri's Game, Star War II, Dinosaure).

Malgré l'apport considérable des surfaces de subdivision au domaine de la modélisation d'objets 3D, plusieurs problématiques restent encore ouvertes : la combinaison de plusieurs surfaces de subdivision avec des opérations booléennes de type CSG pour modéliser des objets complexes. Cette opération nécessite un calcul des courbes d'intersection entre les surfaces à combiner. les problèmes liés à l'introduction d'un espace paramétrique non uniforme, de sommets de valence quelconque et de contraintes de discontinuité.

Nos objectifs dans le cadre de la thèse de Sandrine Lanquetin se résument en : étudier les algorithmes permettant de trouver une approximation des courbes d'intersection entre surfaces de subdivision et d'utiliser cette approximation pour effectuer des combinaisons d'objets (unions, intersection, différences).

S. Lanquetin, S. Foufou, H. Kheddouci et M. Neveu, « A Graph Based Algorithm For Intersection Of Subdivision Surfaces », *In Proceedings of ICCSA'03*, Kumar, Gavrilova, Kenneth Tan and L'Ecuyer Editors, LNCS 2669, Springer-Verlag, Montréal, Canada, 2003.

S. Lanquetin, S. Foufou, H. Kheddouci et M. Neveu, « Computing Subdivision Surface Intersection », *Proceedings of the Int'l Conf. WSCG'03*, Plzen, République Tchèque, 3-7 février 2003, pp 73-76.

S. Lanquetin, S. Foufou, H. Kheddouci et M. Neveu, « Deux algorithmes d'intersection de surfaces de subdivision », *Actes des journées AFIG'02*, Lyon, Décembre 2002, pp 251-258.

S. Lanquetin, S. Foufou et M. Neveu, « Intersection des surfaces de subdivision », *Actes des journées du GTMG'02*, Nantes, 27-28 mars 2002, pp 86-95.

• IRENav, Brest

Les activités de recherche du groupe SIG de l'IRENav concernent le domaine des Systèmes d'Information Géographique et leur application aux domaines marins. Les techniques de détection des intersections et des collisions sont utilisées notamment en cartographie numérique (détection des conflits réels et visuels entre courbes de niveau lors de la construction des cartes marines afin de garantir la lisibilité de la carte en fonction de son échelle) et pour le développement d'outils d'aide à la navigation (détection des collisions avec le relief ou d'autres navires pour la modélisation du transport maritime). Il s'agit notamment de mettre en place des méthodes de détection rapides et robustes pour connaître l'existence d'une intersection (critères de précision).

L'objectif n'est pas la localisation exacte du conflit mais sa caractérisation afin de permettre le choix de méthodes de correction adaptées.

- CEA, Fontenay–aux–Roses

Travaillant sur des simulations de montage / démontage ou d'opération de maintenance à partir de maquettes numériques et avec un retour haptique, nous sommes confrontés à la présence d'objets déformables dans les scènes (enclipsages, cablages...) Pour pouvoir interagir avec des objets déformables de façon interactive, il faut à la fois une détection de collision rapide, relativement précise sur la localisation des contacts et pouvant s'adapter aux changements de forme des objets.

C. Duriez, C.Andriot, A. Kheddar, « Interactive haptics for virtual prototyping of deformable objects : snap-in tasks case », *Proceedings of Eurohaptics 2003*, Dublin, 2003.

- SimTeam, Paris

SimTeam, distributeur et prestataire de services autour des technologies de l'haptique (entre autres), développe depuis plusieurs années des algorithmes permettant de toucher des modèles de grande complexité. Après avoir travaillé sur le rendu haptique point-surface, SimTeam se consacre fortement sur l'interaction objets/objets avec retour d'effort. Ceci implique l'utilisation d'algorithmes de calcul des collisions extrêmement performants. Nous nous lançons dans une phase de recherche et développement de 3 ans, à l'issue de laquelle nous aurons des outils permettant d'adresser des problèmes du domaine industriel de manière adéquate.

Chapitre 5

Résultats de l'AS et Perspectives

5.1 Résultats

Le résultat immédiat de l'AS est principalement constitué par l'état de l'art auquel de nombreux membres de l'AS ont contribué. Comme indiqué dans l'introduction de ce document, les techniques sont si diverses et issues de communauté différentes qu'un travail de fond pour rassembler ses approches était nécessaire.

D'autres résultats sont cependant attendus dans les mois à venir, car certaines collaborations possibles ont été mises en évidence : par exemple, le LERI est demandeur de méthode de détection de collisions exactes pour les objets qu'ils simulent, le LIFL désire enrichir ses simulation avec des calculs de réponses plus proches de la réalité, le LSC, le LIFL, SIMTEAM et le CEA étudient des méthodes de détection de collision efficaces pour le retour d'effort et cherchent des méthodes pour calculer les réactions aux collisions avec les corps déformables... Il est ainsi apparu clairement que les demandes des laboratoires pouvaient trouver écho dans les autres laboratoires de l'AS.

Un autre résultat envisagé est de rassembler les états de l'art et les travaux présentés de cette AS dans un ouvrage. Des contacts avec un éditeur ont d'orès-et-déjà été pris.

5.2 Conclusion et Perspectives

Dans ce document, nous avons fait état d'un certain nombre de problèmes que posent encore de nos jours la détection des collisions. Une part de ces problèmes sont par ailleurs l'objet de recherches au niveau international et même français. Nous n'avons certes jamais fait mention de verrous technologiques. En effet, de nombreux utilisateurs de la détection de collision se contentent des solutions actuelles, même imparfaites, ou contournent les difficultés par des compromis, qui se révèlent toujours limitatifs et manquent souvent de robustesse.

Cette réelle insatisfaction de l'existant a largement contribué à l'intérêt que cette AS a suscité en France. Les communautés d'Animation, de Réalité Virtuelle, de Robotique et de Modélisation Géométrique ont pu apporter à cette AS leur contribution respective. Il a été mis en évidence le besoin d'un standard. La détection des collisions est un outils utile pour diverses applications, comme le rendu projectif l'est pour la synthèse d'images temps réel. Cette comparaison va plus loin encore. À l'image d'OpenGL, bibliothèque de rendu graphique standardisée mais suffisamment souple pour s'adapter aux diverses applications, la détection de collision a besoin d'une bibliothèque véritablement générale, mais réglable selon les besoins. Comme OpenGL, cette bibliothèque se doit de pouvoir donner les spécifications de cartes électroniques dédiés à la détection de collisions. L'élaboration de ce standard est à notre avis l'enjeu majeur des recherches

en détection et traitement des collisions pour les années à venir.

Chapitre 6

Bilan financier

Crédits LIFL/LERI		10 000,00 €
MESEURE Philippe	Mission septembre 2002 Lausanne	-806,69 €
MESEURE Philippe	Mission du 05/02 -06/02/03 Grenoble	-193,71 €
CAZIER David	Mission du 05/02 -06/02/03 Grenoble	-231,88 €
REMION Yannick	Mission du 05/02 -06/02/03 Grenoble	-239,22 €
FOUFOU Sebti	Mission du 02/05/03 Villeneuve d'ascq	-132,92 €
CAZIER David	Mission du 01/05-03/05/03 villeneuve d'ascq	-88,27 €
REMION Yannick	Mission du 02/05/03 villeneuve d'ascq	-116,20 €
MESEURE Philippe	Inscription au GTAS'03	-74,00 €
MESEURE Philippe	Mission du 16/06-18/06/03 Brest	-297,98 €
CAZIER David	Inscription au GTAS'03	-86,00 €
CAZIER David	Mission du 16/06-18/06/03 Brest	-276,42 €
MESEURE Philippe	Mission du 29/10/03 villeneuve d'ascq-Evry	-111,40 €
FAURE François	Mission du 29/10/03 Evry (Paris)	-152,72 €
MESEURE Philippe	Mission du 11 au 13/11/03 villeneuve d'ascq	-204,72 €
IRISA	Inscription AFIG 03 (5 personnes)	-247,00 €
LUCAS Laurent	Mission du 03 au 05/12/03 Paris	-161,10 €
REMION Yannick	Mission du 03/12/03 Paris	-39,38 €
BENASSAROU A.	Mission du 03 au 05/12/03 Paris	-39,38 €
JONQUET Antoine	Mission du 03 au 05/12/03 Paris	-39,38 €
GILLARD Didier	Mission du 04 au 05/12/03 Paris	-39,38 €
Invité AFIG	Mission du 01 au 06/12/03 à Paris	-1 396,80 €
Disponible		5 025,45 €
Crédits LSC		4900,00 €
A. Kheddar	Mission 6/02 Grenoble	-249,59 €
A. Kheddar	Mission 16-18/06 Brest	-261,03 €
S. Redon	Mission 16-18/06 Brest	-562,59 €
	Equipement (H.T.)	-3800,00 €
Disponible		26,79 €
Crédits GRAVIR		4900,00 €
	Fonctionnement	-705,32 €
	Missions	-4088,32 €
Disponible		106,36 €

Crédits IRISA		4900,00 €
Disponible	A Compléter	€
Crédits LIRMM		4900,00 €
	Fonctionnement	-900,00 €
	Equipement	-2665,00 €
	Missions	-1244,00 €
Disponible		91,00 €
Crédits IRCCYN		4900,00 €
	Prélèvement laboratoire 15%	-735,00 €
	Mission Grenoble	-701,00 €
	Equipement	-2600,00 €
	Equipement	-864,00 €
Disponible		0 €

Deuxième partie

Annexe Technique

Introduction

Cette partie se constitue essentiellement des états de l'art élaborés durant l'AS. Elle aborde successivement les aspects de détection exacte, détection approximative, calcul de la réponse puis applications. Elle s'adresse aux spécialistes ou aux chercheurs désireux d'avoir une synthèse de l'existant en algorithmique de détection de collision.

Le premier chapitre est dédié à la détection « exacte » de collision. Il s'agit de ce que l'on a convenu d'appeler « noyaux ». La DdC ou de proximité s'effectue entre des paires de primitives géométriques diverses. Les propriétés de convexité permettent d'exploiter une panoplie d'outils mathématiques (espaces convexes et leurs propriétés) pour formaliser analytiquement le problème de la DdC. Ces outils sont associés assez souvent à des idées originales pour exclure une collision ; on parlera de plan séparateur, de distances signées, etc. La DdC entre la primitive élémentaire : « le triangle » est aussi abordé ainsi que quelques approches s'adressant aux surfaces paramétriques. On aborde aussi la DdC en mode continu, i.e. par interpolation de la trajectoire et on présente les algorithmes qui ont été proposée pour cette approche.

Le deuxième chapitre est dédié aux procédés d'accélération de la DdC. Cette phase est essentielle pour éliminer les cas où un traitement entre primitives de bas niveau est inutile. On aborde les techniques de hiérarchisation par volumes englobants. Les volumes les plus utilisés sont présentés et comparés au vu de leur rapidité et de la complexité du recouvrement. On y trouve aussi les techniques d'accélération exploitant la carte graphique (i.e. le *hardware*), elle suscitent ces derniers temps beaucoup d'intérêt dans la communauté car la programmation est facilité et, comme on se base sur le *hardware*, le traitement semble très rapide. Enfin nous présentons les considérations supplémentaires à prendre en compte en vu de l'extension des approches accélératrices à des modes de détection continus. En effet, pour des raisons de cohérence évidentes, il ne peut être envisagé d'associé des méthodes d'accélérations discrètes à des noyaux de nature continue.

Le troisième chapitre est dédié aux algorithmes de réponse à la collision. Seules les approches corps rigides sont présentées. Après des rappels en cinématique, en dynamique du mouvement et les modèles de frottement, diverses approches en terme de réponse sont présentés. On y trouve les approches par pénalités, par contraintes, par impulsions ainsi que des optimisations diverses. Il est clair que la fonction et la nature de la DdC sont extrêmement liées à l'algorithmique de « réponse » ou tout simplement au moteur d'animation dynamique envisagé.

Le dernier chapitre donne la liste des algorithmes et logiciels de détection de collision existants ainsi que leurs caractéristiques et spécificités sous forme d'un tableau comparatif. Les problèmes liés à leur intégration à l'infohaptie (informatique du rendu haptique) sont discutés.

La partie annexe technique se termine par une bibliographie.

Chapitre 7

Détection exacte des collisions entre primitives

La mise en œuvre de l'algorithmique de détection de collision dépend du modèle géométrique utilisé pour représenter la géométrie des objets virtuels. Dans le cas d'une représentation à base de polygones, la détection de collision se fait entre primitives géométriques simples : sommet, segment, face. Cette représentation offre l'avantage d'être assez générique, et constitue la base commune des librairies graphiques existantes. Le problème de cette représentation réside dans le fait que sa précision dépend de la discrétisation, notamment lorsque les objets sont courbés. Dans le cas d'une représentation par CSG (constructive solid geometry), très utilisée en industrie et bureaux d'études CAO, la détection de collision gagne à être effectuée sur la base de formes géométriques élémentaires (cylindre, cube...) car c'est sur cette base que les objets virtuels sont constitués/modélisés (par des opérations d'union et de soustraction de primitives géométriques). En industrie, mais aussi dans beaucoup d'autres applications, les surfaces implicites et NURBS sont utilisées pour construire et paramétrer la forme géométriques des objets. En disposant de modèles analytiques, la détection de collision peut s'effectuer sur la base de fonction d'appartenance. Enfin dans d'autres applications, les éléments de volume cubique sont à la base de la représentation de la scène virtuelle, notamment lorsqu'il s'agit d'animer des gaz, des fluides ou de visualiser des données issues de l'imagerie médicale, les algorithmes de détection de collision peuvent alors être simples mais coûteux en temps de calcul.

Nous présentons ici divers algorithmes permettant de préciser si deux objets entrent en collision. Ces approches sont classifiées par types de primitives en critère principal et par type de détection (détection d'intersection vide, calcul d'interpénétration ou détection de contact) en critère secondaire. Les types de primitives que nous considérons ici sont les polyèdres convexes suivis par le cas général des polyèdres quelconques, le cas des objets définis par des surfaces implicites puis par des surfaces paramétriques.

7.1 Détection entre polyèdres convexes

7.1.1 Détection d'intersection vide

Les algorithmes permettant de détecter si l'intersection entre deux polyèdres est vide ou non cherchent à séparer les deux polyèdres : soit on trouve un plan tel que les deux objets soient de part et d'autre du plan, soit on calcule une borne minimale de la distance les séparant. On souhaite en outre que cette technique de séparation n'échoue que lorsque les deux objets se recouvrent.

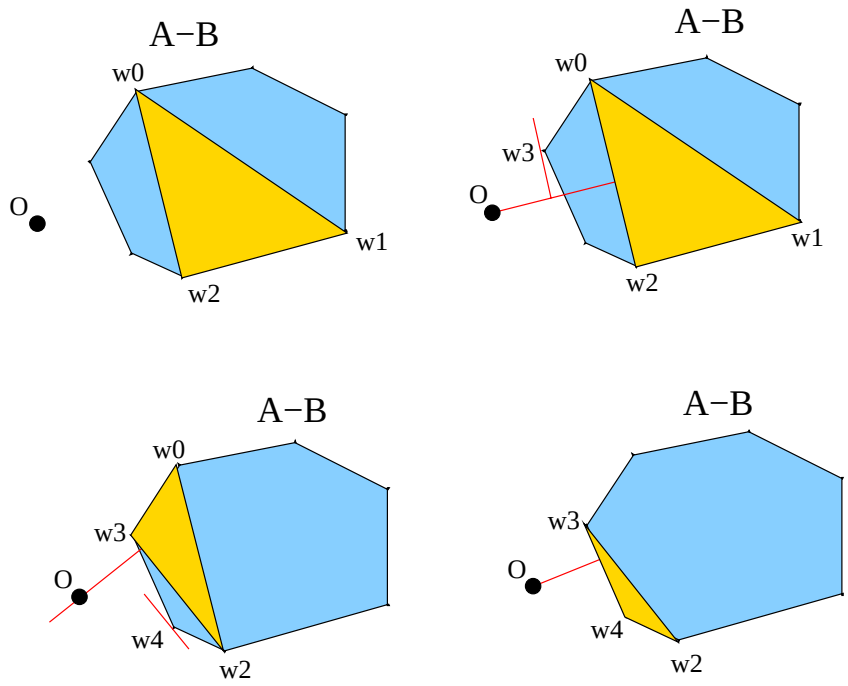

FIG. 7.1 – Algorithme GJK. En cyan, la différence de Minkowski des deux polyèdres, dont on cherche la distance par rapport à l'origine. À chaque itération, on calcule la distance du simplexe en jaune par rapport à l'origine. Cette distance fournit une direction. On cherche alors le point support du polyèdre dans cette direction afin de former un nouveau simplexe.

- Calcul de distance inter-polyèdres

Trois grandes approches ont été proposées pour évaluer, soit précisément, soit par borne inférieure, la distance entre deux polyèdres convexes. La première approche (dite GJK : voir figure 7.1) se base sur la différence de Minkowski des polyèdres [CC86], que l'on approxime par des simplexes [GJK88] pour éviter sa construction explicite. Ce type d'algorithmes a connu diverses améliorations (EGJK [Cam97], ISA-GJK [Ber99], MS [SdM00]). La seconde approche [DK90] (dite DK, voir figure 7.2) consiste à construire une suite d'approximations de plus en plus grossières des polyèdres pour évaluer par itération la distance. Enfin la troisième approche [LC91], baptisée plus tard Voronoï Marching, se base sur une découpe de l'espace environnant chaque polyèdre en régions de Voronoï (voir figure 7.3). Diverses améliorations ont là aussi été proposées [PML95, Mir97].

- Recherche de plans séparateurs

Lorsque les deux objets sont situés de part et d'autre d'un plan, on peut conclure qu'ils ne s'intersectent pas. Les approches possibles construisent ce plan de façon différente : en calculant une borne minimale de la distance de séparation [Ber99] ou par produit vectoriel [CW96]. Lors d'un mouvement, [Bar90] propose d'exploiter la cohérence temporelle en testant d'abord le plan calculé au pas précédent. De ce fait, la recherche des plans séparateurs est généralement plus rapide que les méthodes d'évaluation des distances.

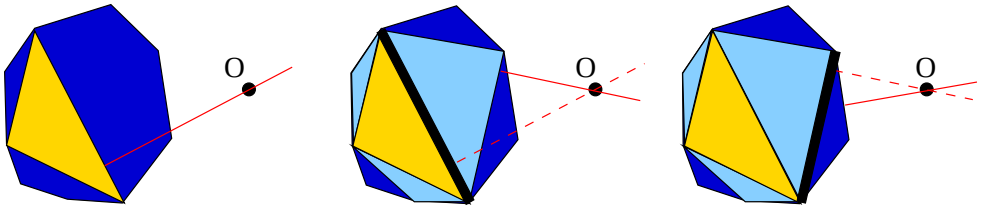

FIG. 7.2 – Algorithme DK. Au départ, l'algorithme se base sur un simplexe approximant le polyèdre en exploitant certains de ses sommets. À chaque étape, on ajoute des sommets du polyèdre permettant d'étendre le simplexe de départ en direction de l'origine.

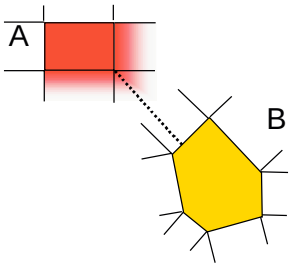

FIG. 7.3 – Algorithme LC-Voronoï Marching. On cherche le couple d'élements de l'objet tel que l'un des éléments est dans la région de Voronoï associée à l'autre élément et réciproquement.

7.1.2 Calcul d'interpénétration

Pour une direction donnée, la distance d'interpénétration entre deux polyèdres convexes ayant n et m sommets peut se calculer, grâce à la hiérarchie de [DK90], en une complexité moins que linéaire en $\mathcal{O}(\log(n) \times \log(m))$ [DHKS93]. La distance d'interpénétration est cependant définie comme l'amplitude de la plus petite translation permettant de séparer deux objets. Le problème est donc de trouver la direction de cette translation. Plusieurs algorithmes se basent sur le GJK pour la calculer ([Cam97, JSL99] et l'*Expanded Polytope Algorithm* (EPA) [Ber01]) ou étendent le voronoï Marching [GME+00]. Une autre approche [AGHP+00] a une complexité inférieure au cas linéaire mais est moins robuste. Enfin, une dernière méthode passe dans l'espace dual (espace des normales) pour trouver la direction de la translation (méthode *DEEP*) [KLM02].

7.1.3 Construction de l'intersection

La construction de l'intersection entre deux polyèdres est souvent très lourde et rarement nécessaire. En géométrie algorithmique, la construction de l'intersection de deux polyèdres convexes peut se faire avec une complexité linéaire, mais ce type d'algorithmes étant difficile à mettre en œuvre, d'autres approches sont proposées, notamment celles utilisant les partitions binaires de l'espace (BSP : Binary Space Partionning, voir [FDFH90]) [TN87, NAT90, Van91].

7.2 Détection entre polyèdres quelconques

Les algorithmes ci-dessus sont restreints au cas convexe. Lorsqu'un polyèdre est concave, l'utilisation des algorithmes précédents sur son enveloppe convexe permet souvent de conclure à la non-collision. La bibliothèque Qhull[1] peut être employée pour la construction de cette enveloppe.

[1] Voir http ://www.geom.umn.edu/software/qhull/

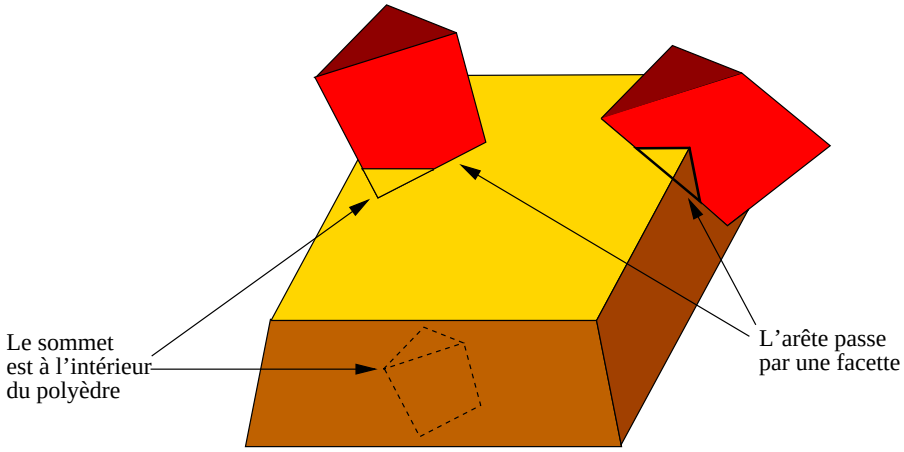

FIG. 7.4 – Détection spatiale entre polyèdres. Dans chaque cas d'intersection, un sommet d'un polyèdre est situé dans l'autre ou une arête passe par une face. Il est courant d'avoir ces deux propriétés vérifiées en même temps, mais ce n'est pas obligatoire.

Cependant, pour définir complètement une collision, des algorithmes spécifiques sont nécessaires.

7.2.1 Détection d'intersection vide

- Calcul de distance inter-polyèdres

Dans le cas concave, seules certaines approches [Qui94, JC98, EL01] se basent sur une hiérarchies de volumes englobants. En parcourant simultanément les hiérarchies des deux objets, on génère une suite croissante convergeant vers la distance.

- Recherche de plans séparateurs

On cherche un plan tel que les points de l'un des objets soit du côté positif, et les points de l'autre objet du côté négatif [HDLM96]. Ceci s'exprime sous la forme d'un problème d'optimisation, qui se résout soit par une programmation linéaire efficace [Sei90] soit par un recuit simulé [Zac01].

7.2.2 Détection des surfaces en intersection

L'intersection de deux polyèdres est non vide si et seulement si l'une des deux conditions suivantes est vérifiée :

– un sommet de l'un appartient à l'autre ;
– une arête de l'un coupe une face de l'autre.

Ces deux conditions sont souvent simultanément vérifiées, mais pas nécessairement (voir figure 7.4). Suivant la stratégie, l'ordre dans lequel les tests sont effectués est différent [Boy79, MW88]. Une autre solution consiste à inspecter les triangles des deux objets entre eux [Möl97] (voir figure 7.5).

Malheureusement, dans ces algorithmes, c'est lorsque tous les tests auront été effectués que l'on pourra conclure qu'il n'y a pas collision. Leur complexité est donc quadratique.

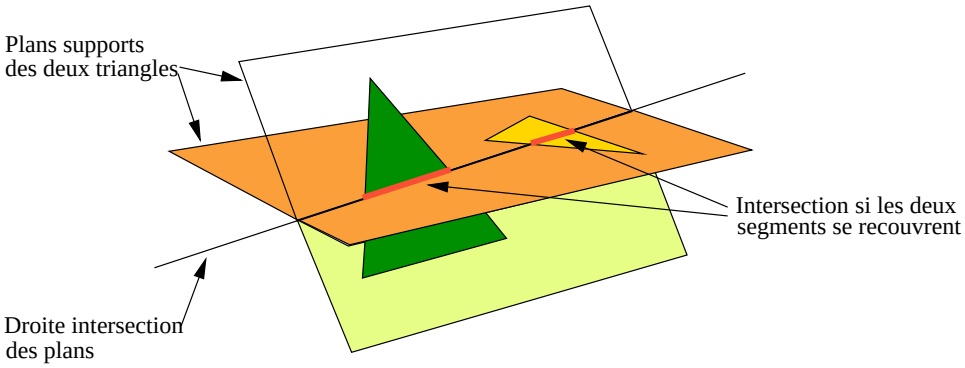

FIG. 7.5 – Intersection spatiale entre triangles

7.2.3 Calcul d'interpénétration

Les algorithmes utilisés dans les cas convexes ne sont pas adaptables aux cas concaves, dont les collisions peuvent se répartir entre plusieurs composantes non connexes. [DHKS93] rappellent que, dans ce cas, la complexité de la construction explicite de la différence de Minkowski est en $\mathcal{O}(n^6)$. D'autre part la notion de distance d'interpénétration est une notion globale. Pour certaines applications, il vaut mieux avoir la distance d'interpénétration de chaque zone d'intersection, lorsque cette dernière n'est pas connexe.

Signalons cependant que [KOLM02b] utilisent une hiérarchie de composantes convexes (mélange des méthodes [KLM02] et [EL01]). Par ailleurs, des approches [EHK+00, FL01] proposent un calcul de l'interpénétration basé sur la carte de profondeur. Malheureusement, les temps indiqués par toutes ces études restent trop élevés.

7.3 Détection entre objets définis par fonctions implicites

Plusieurs auteurs préconisent l'emploi d'objets volumiques dont la surface est définie par équations implicites. En effet, ce formalisme procure un moyen simple permettant de décider si un point est à l'intérieur ou non de l'objet. De façon générale, la surface est définie par une équation du style $F(x, y, z) = 0$. Les points se situant à l'intérieur sont les seuls à vérifier $F(x, y, z) > 0$.

7.3.1 Détection d'intersection vide

[SP95] remarquent que si la borne supérieure de la fonction $min\{F, G\}$ est positive, alors il existe une intersection et proposent une méthode pour trouver cette borne.

7.3.2 Calcul de distance

[LM95] cherchent les deux points permettant d'évaluer la distance minimale entre les deux surfaces. Le problème revient à trouver A sur la surface F et B sur la surface G tels que :

$$\begin{aligned} F(A) &= 0 \\ G(B) &= 0 \\ \vec{\nabla}F(A) &= \lambda\vec{\nabla}G(B) \\ \vec{AB} &= \mu\vec{\nabla}G(B) \end{aligned} \tag{7.1}$$

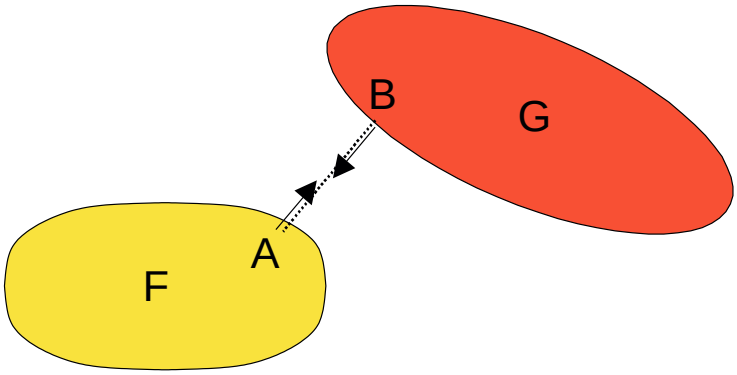

FIG. 7.6 – Calcul de distance entre surfaces implicites

A, B, l et m sont les inconnues (voir figure 7.6). Une approximation locale en polygones permet de trouver les solutions approchées, servant de conditions initiales à la résolution itérative du système non linéaire à résoudre.

7.3.3 Détection de parties d'objets

Pour détecter si deux surfaces implicites se coupent, il faut échantillonner la surface de l'un des objets puis inspecter si au moins un des points obtenus se trouve à l'intérieur de l'autre volume. Il faut ensuite vérifier le cas où le second objet est complètement inclus dans le premier.

Les approches permettant la transformation d'une surface implicite en faces polygonales peuvent être employées à bon escient pour l'échantillonnage de la surface. Cependant, comme les liens topologiques entre points de l'échantillonnage ne sont pas nécessaires, des méthodes comme l'échantillonnage à base de grains [DCG94] exploitant la cohérence temporelle seront plus efficaces que l'algorithme des *Marching Cubes* [LC87], par exemple. Dans le cas particulier des superquadriques, [SP91] ont recours à une paramétrisation de la surface permettant un échantillonnage immédiat.

7.3.4 Construction de l'intersection

Par la méthode consistant à chercher les points de la surface d'un objet A présents à l'intérieur d'autre objet B, on peut définir un ensemble de points situés à la surface de l'intersection. Une approximation polyédrique de l'intersection peut alors être construite par triangulation par exemple. Il faut cependant prendre garde aux cas d'intersections non connexes.

7.4 Détection entre surfaces paramétriques

La détection des intersections entre surfaces complexes est généralement impossible à effectuer de façon formelle. Souvent, on procède par approximation, en subdivisant la surface et en approximant les morceaux par des volumes englobants simples [HBZ90, HDLM96].

En deux dimensions, de nombreuses approches géométriques permettent par itération de déterminer si deux courbes splines s'intersectent. Les méthodes sont classifiées en méthodes par subdivision [LR80] (version 2D des méthodes utilisées pour les surfaces en 3D), mais également en méthodes par découpage [SN90] ou par réduction d'intervalles (pour les B-splines [Dan92]).

Des méthodes ont également été proposées pour calculer la distance entre les surfaces. De façon similaire à l'équation (7.1), [LM95] posent le système permettant de trouver les points impliqués dans la distance minimale :

$$\vec{F}(s_0,t_0) - \vec{G}(u_0,v_0) = \lambda \left(\frac{\partial \vec{G}}{\partial u}(u_0,v_0) \wedge \frac{\partial \vec{G}}{\partial u}(u_0,v_0) \right)$$
$$\left(\frac{\partial \vec{F}}{\partial s}(s_0,t_0) \wedge \frac{\partial \vec{F}}{\partial t}(s_0,t_0) \right) = \mu \left(\frac{\partial \vec{G}}{\partial u}(u_0,v_0) \wedge \frac{\partial \vec{G}}{\partial u}(u_0,v_0) \right)$$

Ce qui nous donne six équations pour les six inconnues que sont s_0, t_0, u_0, v_0, λ et μ. Là aussi, une approximation locale de la surface en polygones est utilisée afin de fournir des conditions initiales à une résolution itérative du système obtenu.

La plupart des méthodes s'attachent à calculer précisément ou approximativement la distance entre deux carreaux de surface. L'algorithme GJK a été adapté à la gestion de ce type d'objets [Ber99, TC98]. Mais les méthodes à base de hiérarchies de volumes sont ici aussi très utiles [JC98, TTRC00]. Citons enfin l'approche de [SWF+93] qui s'intéresse aux collisions multi-points entre surfaces complexes. Pour cela, une approche par contraintes résolues par l'arithmétique d'intervalles est proposée.

7.5 Détection spatio-temporelle discrète

Il est possible d'utiliser les algorithmes purement 3D vus ci-dessus pour trouver l'instant de contact entre deux objets.

Une première approche procède par dichotomie, en se basant sur un algorithme à réponse purement booléenne. Lorsqu'à deux instants donnés, la réponse est différente, c'est que le contact est survenu à un moment intermédiaire. On découpe alors l'intervalle en deux parties identiques et on procède à un nouveau test. On retient alors l'intervalle où la réponse est différente aux deux bornes. L'algorithme est alors réitérer, jusqu'à obtenir un intervalle petit, permettant une approximation aussi fine que voulu de l'instant de contact.

Une autre solution nommée *backtracking* consiste à se baser sur une mesure de la profondeur d'interpénétration et la vitesse des objets pour remonter à un temps de contact approximatif ([Hah88] dans le cas rigide). Dans le cas des objets déformables, on peut uniquement ramener la partie en intersection en une situation de contact, par un mécanisme de projection [Pro97].

Un dernière solution consiste à anticiper le moment de la collision : à partir d'une borne minimale de la distance et une borne maximal de la vitesse, on obtient un temps minimal pendant lequel aucune collision n'est possible [CK86, LC92]. Par exemple [GH89] proposent de projeter les objets sur la ligne les séparant (cette ligne est obtenue grâce à un GJK), et, en prenant en compte les vitesses relatives, d'anticiper le moment où les projections doivent s'intersecter. La méthode est alors appliquée itérativement jusqu'à ce que l'algorithme GJK conclue à la collision.

Cependant, quelque soit l'approche, il est impossible de garantir que toutes les collisions sont détectées. En effet, il peut advenir que des objets passent complètement l'un au travers de l'autre entre deux inspections. Les algorithmes discrets sont incapables de se rendre compte de l'interpénétration. En pratique, si Δt est l'intervalle de temps entre deux inspection, et $Vmax$ la vitesse relative des deux objets, des interpénétrations d'une longueur inférieure à $lmax = Vmax \times \Delta t$ peuvent ne pas être détectées.

7.6 Détection spatio-temporelle semi-continue et continue

La majorité des simulateurs dynamiques, notamment ceux qui sont utilisés pour des applications de réalité virtuelle utilisent des algorithmes de DdC discrets. Peu de méthodes de détection de collisions continues sont utilisées si bien que les approches ne sont pas aussi nombreuses que dans le cas discret.

Pour ce qui est des méthodes continues, dites aussi détection par interpolation, le principe général consiste à :

1. interpoler la trajectoire de l'objet entre deux instants discrets successifs ;
2. paramétrer l'interpolation en temps (c'est la raison pour laquelle elle sont parfois appelées méthodes 4D) ;
3. déterminer le « premier » temps t_c pour lequel soit deux objets virtuels entrent en collision ou qu'un objet virtuel se touche.

Notons toutefois que ce qui caractérise un algorithme de DdC continue c'est l'hypothèse suivante :

Hypothèse 1 *Si pour une interpolation de trajectoire donnée, une collision est détectée dans l'intervalle de temps* $[t_k, t_{k+1}]$ *alors il y a effectivement une collision.*

Cette hypothèse n'est que *théoriquement* fausse, dans le cas où la trajectoire interpolée n'est pas la trajectoire réelle de l'objet. Ce qui pourrait être le cas quand la trajectoire entre deux pas discrets successifs n'est pas définie. Par exemple, lorsque l'objet virtuel est manipulé par l'opérateur, le moteur de simulation ne prélève la trajectoire qu'à des pas discret. Il suffit donc de choisir une méthode d'interpolation qui génère une trajectoire qui ne s'éloigne pas trop de la trajectoire réelle (quand elle est connue). Ainsi, on peut considérer que cette hypothèse est toujours vérifiée.

Plusieurs approches ont été proposées. Dans [Can86], Canny utilise une paramétrisation des trajectoires d'objets polyédriques basée sur des quaternions. La détection de collisions est formulée par l'intermédiaire de polynômes dont la résolution (recherche de zéros) détermine le temps de collision. Cameron [Cam90] présente une méthode d'extrusion spatio-temporelle pour déterminer des contacts entre des objets CSG (*constructive solid geometry*) mobiles (voir [FDFH90], chapitre 12), et la détection revient à détecter si l'arbre CSG défini par l'intersection des deux hypervolumes est vide. D'autres méthodes utilisent des bornes de Lipschitz et une méthode de dichotomie pour trouver l'instant de premier contact entre des surfaces paramétrées dépendant du temps. Duff [Duf92] emploie l'arithmétique d'intervalles et une méthode de dichotomie pour détecter des collisions entre des combinaisons booléennes de surfaces implicites. Snyder *et al.*[SWF+93] utilisent l'arithmétique d'intervalles et des méthodes de Newton par intervalles pour accélérer significativement ces approches. Dans [Bar90], Baraff se base sur une version temporelle de l'équation (2.1) afin de trouver l'évolution de la distance entre des objets définis par des équations implicites :

$$\begin{gathered} F(A_t, t) = 0 \\ G(B_t, t) = 0 \\ \vec{\nabla} F(A_t, t) + \lambda \vec{\nabla} G(B_t, t) = \vec{0} \\ \vec{A_t B_t} + \mu \vec{\nabla} G(B_t, t) = \vec{0} \end{gathered}$$

La collision est détectée lorsque $A_t = B_t$. Une méthode de résolution de ce système non linéaire est proposée.

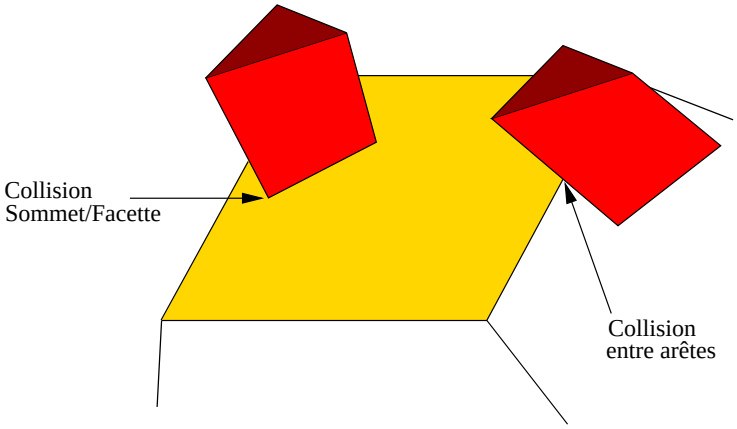

FIG. 7.7 – Détection spatio-temporelle entre polyèdres.

Mirtich [Mir97] n'utilise que des mouvements balistiques et peut borner inférieurement les instants d'impact, ce qui permet de réduire la fréquence des tests de collision. Cependant, les bornes sont difficiles à obtenir pour des corps articulés ou pour des mouvements plus complexes. Aussi la majorité de ces méthodes ne s'appliquent pas forcément aux objets déformables. L'approche de [FH94] considère le volume balayé durant le mouvement comme une union de domaines convexes construits avec une arithmétique d'intervalle utilisable pour la détection des collisions.

[Boy79] montre que lorsque deux objets polyédriques entrent en collision, c'est toujours lorsqu'une arête (incluant ses extrêmités) de l'un des objets entre en collision avec une face de l'autre objet. Deux situations se présentent alors : soit une extrémité de l'arête atteint la surface, soit l'arête touche un des bords de la face (voir figure 7.7). En faisant l'hypothèse de mouvements rigides, le contact est déterminé par des intersections de figures géométriques.

Des approches analytiques sont possibles. On considère tous les couples formés par une face de l'un et une face de l'autre objet. Tout d'abord, la détection n'est utile que lorsque les faces des objets se dirigent l'une vers l'autre, ce que propose de vérifier [Van94] dans le cas des objets rigides. Le principal problème des méthodes à contact et qu'il faut déterminer une trajectoire des objets, alors que leurs positions ne sont connues qu'à des instants discrets. Interpoler revient alors à approximer la trajectoire.

Pour trouver de façon analytique l'instant de la collision, la détection peut être facilitée si les faces des polyèdres sont triangulaires et si on suppose des trajectoires rectilignes uniformes des points. Dans ce cas, il faut définir si un sommet du polyèdre passe par un triangle ou si deux arêtes de triangles se croisent. L'équation d'intersection entre un sommet et une face est alors de degré 1 [MW88] ou de degré 3 [Pro97].

L'étude de trajectoire plus complexe est cependant possible : par itération [GH89], ou en s'appuyant sur une approximation en déplacement rigide. S. Redon [RKC00, RKC02a] propose une interpolation de la trajectoire par un mouvement intermédiaire arbitraire devant toutefois respecter certaines contraintes. Soient t_n et t_{n+1} deux instants discrets successifs quelconques, pour lesquels les positions des objets sont connues, et $I_n = [t_n, t_{n+1}]$ l'intervalle de temps intermédiaire correspondant, pendant lequel le mouvement réel de l'objet n'est pas utilisable. Alors le mouvement intermédiaire arbitrairement choisi doit :

- conserver durant tout le mouvement les paramètres physiques de l'objet, c'est-à-dire si l'objet est rigide alors la distance entre deux points quelconques de l'objet doit être la même durant tout le mouvement ;
- avoir les même positions de l'objet aux instants discrets que le mouvement réel ;

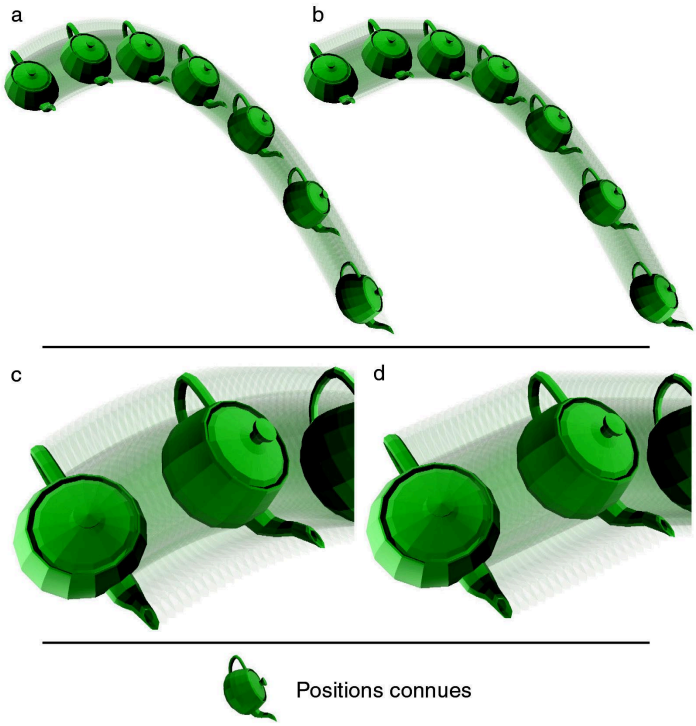

FIG. 7.8 – Utilisation d'un mouvement intermédiaire arbitraire pour interpoler les positions successives connues de la théière (d'après S. Redon).

- passer continûment de la position réelle de l'objet à l'instant t_n à la position réelle de l'objet à l'instant t_{n+1} ;
- être le plus proche possible du mouvement réel quand il est connu. Cette contrainte est difficile à quantifier en l'absence d'informations sur le mouvement réel.

Dans le cas d'un simulateur dynamique interactif qui calcule la position d'un objet et *affiche* un nouvel état de l'environnement à chaque pas de temps. Entre deux états, le mouvement de l'objet n'est pas visible par l'utilisateur et peut être remplacé par un *mouvement arbitraire*. Puisque les positions réelles sont respectées aux instants discrets successifs, seul le mouvement *local* de l'objet est modifié, et son mouvement *global* est conservé. C'est ce qui est illustré sur la figure 7.8 où l'objet en question est une théière.

Le fait de remplacer le mouvement des objets par un mouvement arbitraire entre deux instants discrets successifs t_n et t_{n+1} n'a de conséquence sur la simulation que si une collision intervient entre ces deux instants, pour une paire quelconque d'objets. Si aucune collision n'est détectée sur l'intervalle de temps intermédiaire, les objets sont placés aux positions souhaitées par le calculateur dynamique. En revanche, si une collision est détectée à l'instant t_c entre deux objets, il est nécessaire d'utiliser les mouvements arbitraires pour calculer les positions de *tous* les objets à l'instant t_c, puisque ce sont ces mouvements qui ont été employés pour la détection de collisions.

Comme les algorithmes retournent pour chaque collision détectée le temps t_c il est souhaitable, pour une paire d'objets, de sauvegarder tous les temps de collision obtenus pour chaque paire de primitives testée. On prenant le $\min_i t_{c_i}$ on peut connaître le lieu et la paire concernée par la première collision. Il est donc possible d'avoir l'état (ou la configuration) exact de l'objet au moment exact de la collision (figure 7.9).

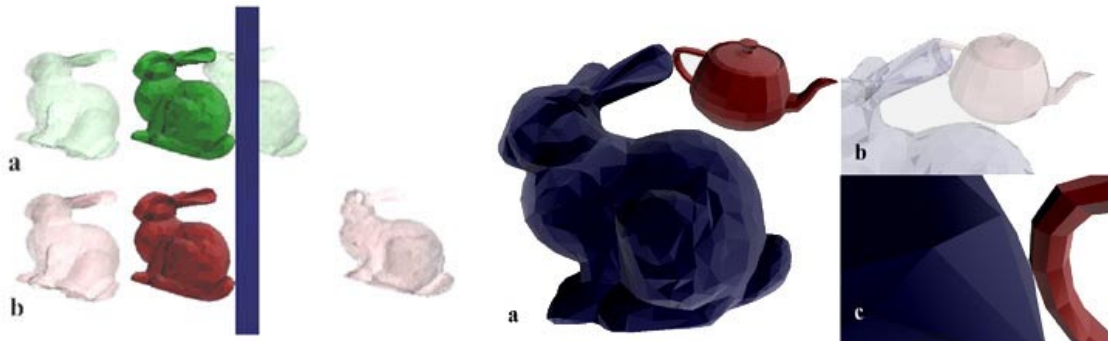

FIG. 7.9 – Avantage de la méthode relativement aux algorithmes discrets. A gauche on peut voir qu'un algorithme discret n'aurait pas détecter la collision (cas b). A droite, l'agrandissement montre la précision de l'algorithme proposé vis-à-vis d'une détection discrète.

7.7 Bilan sur la détection exacte entre objets

Nous avons montré que les divers modèles nécessitaient des algorithmes spécifiques de détection de collision.

La détection de collisions entre objets polyédriques a fait l'objet d'une grande part des recherches. Les communautés de robotique, de géométrie algorithmique et de synthèse d'images ont proposé leurs approches respectives. Les algorithmes les plus efficaces sont ceux basés sur l'hypothèse de convexité. Dans le cas général, il faut souvent recourir à des hiérarchies de volumes ou appliquer des traitements systématiques entre les parties des objets. La complexité des algorithmes devient alors quadratique.

Détecter des intersections peut sembler plus pratique pour les surfaces implicites que pour les modèles polyédriques. En effet, tester l'inclusion d'un point dans la structure nécessite simplement dans le premier cas l'évaluation d'une expression tandis que dans le second cas, il faut parcourir toutes les facettes, arêtes et sommets de l'objet. Cette différence de complexité n'est qu'apparente. Tout d'abord, l'expression d'une fonction implicite peut parfois être très gourmande en temps machine (si elle emploie des expressions trigonométriques ou exponentielles). Ensuite, les fonctions implicites exigent d'être échantillonnées durant la simulation, ce qui devient la partie lente de la détection. De ce fait, il semble plus profitable de tester une surface implicite avec un polyèdre et non pas une autre surface implicite.

Le dernier cas que nous avons exposé est celui des objets définis par surface paramétrique, où les calculs nécessaires sont lourds et les méthodes recourent la plupart du temps à des approximations polygonales.

Signalons que d'autres types de modélisation n'ont pas été abordées ici car elles sont très spécifiques (objets définis sous forme de voxels en géométrie discrète, les CSG...).

Finalement, nous avons également étudié les méthodes de détection continue, qui sont peu nombreuses mais sont les seules à garantir de ne pas omettre de collisions.

Chapitre 8

Accélération de la détection

8.1 Problématique

Dans le paragraphe précédent, nous avons étudié comment on pouvait déterminer la collision entre deux objets. Les algorithmes que nous avons montrés sont coûteux (complexité quadratique). Cependant, les environnements virtuels comportent la plupart du temps de nombreux objets. Le nombre de couples d'objets à examiner est donc $n \times (n-1)/2$, donc une complexité en $\mathcal{O}(n^2)$, où n est le nombre d'objets de la scène.

Des techniques d'accélération sont donc nécessaires, mais sont dépendantes du contexte. Les techniques de type *2-body* s'appliquent aux environnements statiques dans lequel un objet est manipulé : on profite alors du fait que l'environnement est immobile. Lorsque plusieurs objets sont mobiles, les techniques de type *N-body* sont alors nécessaires.

La détection de collision est organisée en pipeline. La première étape dite de *recherche de proximité* (*broad-phase*) cherche à réduire le nombre de couples d'objets à inspecter. Elle fournit un ensemble de couple d'objets en forte probabilité d'intersection. La *broad-phase* est suivie par la *narrow-phase* ou *détection approximative* dont l'objectif est de déterminer rapidement mais pas forcément exactement la collision pour chacun de ces couples. Elle tente simplement de définir les zones de collisions probables ou au contraire les collisions impossibles. Elle permet en pratique de réduire la complexité de la détection exacte.

8.2 La recherche de proximité (*broad-phase*)

Si, par hypothèse, certains objets ne pourront jamais se rencontrer, on peut employer une matrice de collisions, constituée de booléens indiquant si deux objets sont, par définition de l'environnement, susceptibles ou non d'entrer en collision [GSF94]. Dans le cas général, la recherche de proximité a recours à deux types de stratégies :

- découpage de l'espace : on élimine les couples d'objets ne se trouvant pas dans la même portion d'espace ;
- critères topologiques et cinématiques : on considère les distances entre les objets, parfois associées aux vitesses de déplacement. Si des objets sont trop éloignés ou que leur vitesse est trop faible pour qu'ils se rencontrent dans un avenir proche, la collision est impossible.

8.2.1 Stratégies de détection par découpage de l'espace

L'idée de base de ces algorithmes est très simple. Deux objets se situant en des endroits éloignés de l'espace n'ont aucune chance d'entrer en collision dans un avenir proche. En subdivisant

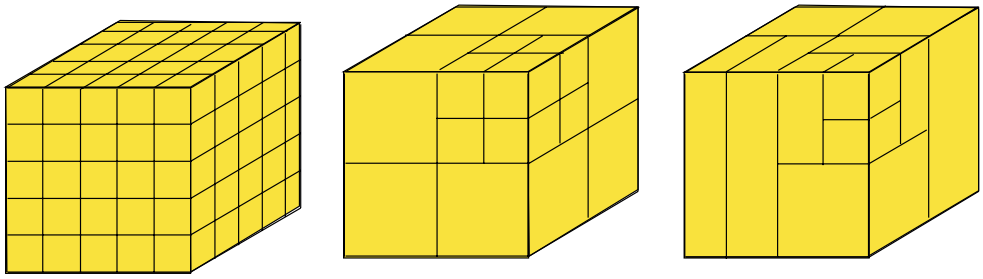

FIG. 8.1 – Divers types de divisions spatiables : grille régulière, voxels et *k-d trees*.

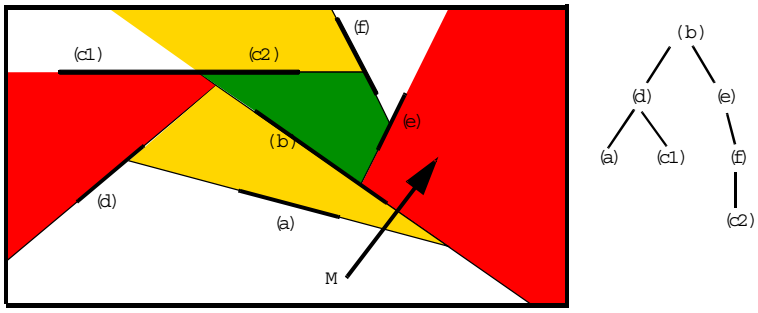

FIG. 8.2 – Algorithme BSP

l'espace, on se fixe des zones bien séparées et on cherche pour chaque objet ou pour chaque partie d'objet dans quelle zone elle se situe. Les partitions de l'espace peuvent se faire d'une façon absolue ou dépendre de la configuration des obstacles immobiles de l'environnement.

Les subdivisions appliquées pour la découpe de l'espace peuvent être indépendantes de l'environnement [BF79]. On fait alors l'analogie avec les techniques de classements de bases de données, et la détection de collision s'apparente alors à une recherche multidimensionnelle par plage (*range searching*) sur ces données. On peut diviser l'espace en une grille régulière [Tur89, Tha00], en une structure hiérarchique de type octrees [Tur89, BT95, SKTK95]. D'autres types de découpages spatiaux sont possibles [HKM95], comme par exemple la structure *k-d tree* [BF79]. La figure 8.1 représente plusieurs exemples de telles subdivisions.

Les subdivisions peuvent également être fournies par l'environnement lui-même, mais cette technique n'est efficace que lorsque l'environnement est statique (contexte de type *2-body*) : les BSP déjà évoqués auparavant sont couramment utilisés dans les jeux vidéo (voir figure 8.2). On peut également utiliser des maillages tétraédriques conformes [HKM95, Gei00] ou un ensemble de contraintes [Can86].

Pour conclure sur les méthodes à subdivision, nous constatons qu'elles permettent des accélérations à la fois entre objets mais également entre primitives d'objet. Cependant, dans les contexte de type *N-body*, les mouvements des objets obligent à mettre à jour la structure ce qui nécessite un temps de calcul additionnel.

8.2.2 Stratégies de détection par topologie et cinématique

Les stratégies *topologiques* s'intéressent au positionnement des objets les uns par rapport aux autres : on cherche alors à supprimer les couples d'objets trop éloignés l'un de l'autre. Les approches *cinématiques* tiennent compte de leur mouvement et les ignorent s'ils ne se dirigent pas l'un vers l'autre ou si leur vitesse ne leur permet pas de se rencontrer.

Certaines approches topologiques se basent sur un classement des objets. On approxime les objets par des boîtes englobantes, toutes définies dans un même repère, dont on cherche les recouvrements. Plusieurs structures ont été proposées, *R-trees* [HKM95] ou d'autres types d'arborescence [OvdS96]. Ces méthodes de complexité de détection moins que linéaire ne sont cependant pas dynamiques (leur construction est de complexité plus que linéaire) ou difficiles à rendre incrémentales (ce qui permettrait de modifier légèrement les structures lors de faibles mouvements des objets). Le *sweep and prune*, où l'on trie les boîtes selon les axes, est généralement l'algorithme le plus employé [CLMP95].

D'autres approches préfèrent calculer les distances entre chaque paire d'objets ou calculer des plans séparateurs. Si la distance tend vers zéro, ou que le plan devient impossible à construire, c'est que les objets entrent en collision. Pour cela, et si nécessaire, on approxime tous les objets par des polyèdres convexes et on modifie les méthodes du paragraphe 7.1 pour qu'elles puissent exploiter la cohérence temporelle : [CW96, Bar90, LC92] et la méthode *SWIFT* [EL00] pour [LC91], [Cam97, Ber99] pour [GJK88], [GHZ99] pour [DK90]. D'autres approximations non polyédriques sont possibles pour le calcul de distance ([JC99] utilisent des sphères).

Enfin, la dernière famille d'approches exploite les vitesses relatives des objets. Ainsi, si les objets s'éloignent, ils ne peuvent entrer en collision [Van94] : c'est le *test du recul* [RKC02b]. Si, en plus des vitesses relatives, les distances inter-objets peuvent être approximativement connues, il est possible d'estimer le temps d'impact [CK86, LC92, Dwo94, DZ93, KGS98].

En conclusion, nous pouvons dire que les méthodes topologiques sont plutôt adaptées aux contextes *N-body*.

8.3 La détection approximative (*narrow-phase*)

L'objectif de la détection approximative est de déterminer l'ensemble (éventuellement vide) des zones de collision afin d'accélérer la détection exacte. Elle débute généralement par des méthodes moins grossières que dans la recherche de proximité et permettant de déterminer si deux objets ont une grande probabilité de se couper, ou si, au contraire et de façon sûre, ils ne sont pas entrés en collision. Les stratégies se basent essentiellement sur l'utilisation de volumes simples qui englobent ou au contraire sont englobés par l'objet et dont les tests de collisions sont plus rapides à effectuer. Par ailleurs, certaines approches exploitant le matériel graphique ont été proposées.

8.3.1 Stratégies de détection par volumes d'approximation

L'utilisation de volumes approximant les objets permet diverses détections. Détecter les non-collisions au moyen de volumes englobants ou conclure à la collision sûre au moyen de volumes englobés [ZPG95]. On peut également utiliser des volumes d'approximation pour calculer des distances inter-objets (si on désire une détection avec un seuil de distance minimale) [Qui94, JC98, JC99, EL00, EL01].

Les volumes utilisés sont les objets géométriques simples comme les sphères, les boîtes de même orientation c'est-à-dire alignées sur des axes (AABB : Axis-Aligned Bounding Boxes), les boîtes à orientation quelconque (OBB : Oriented Bounding Boxes). Cependant, d'autres volumes plus complexes sont exploitables comme les polytopes à orientation discrète (k–DOP), les enveloppes convexes des objets, les Quantized Orientated Slabs with Primary Orientations (QuOSPO) [He99], les spherical-Shells [KPLM98]. La figure 8.3 et le tableau 8.1 fournissent les tests de détection de collision pour des volumes couramment employés.

Volumes	Tests de non collision
Sphères	$\text{dist}(O1, O2)^2 > (R1 + R2)^2$
AABB	$\exists k \in \{x, y, z\}, P_{\max}[k] < P_{\min}[k]$
k–DOP	$\exists k \in \{1 \cdots l\}, P_{\max}[k] < P_{\min}[k]$
OBB	$\exists \vec{L} \in \{15\ axes\}, \lvert\vec{T}.\vec{L}\rvert > \sum_{i=1}^{3} \lvert a_i \vec{A}_i.\vec{L}\rvert + \sum_{i=1}^{3} \lvert b_i \vec{B}_i.\vec{L}\rvert$
Enveloppe ou autre volume convexe	GJK, DK ou LC-Voronoï Marching

TAB. 8.1 – Volumes englobants et tests de collision.

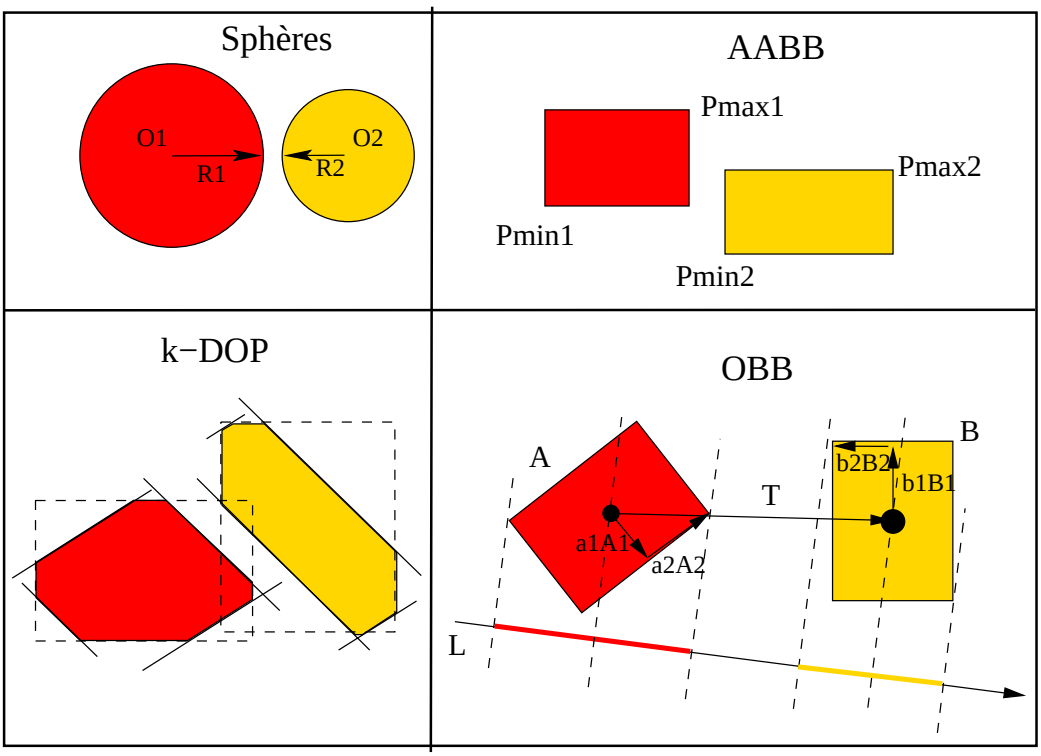

FIG. 8.3 – Test de collision entre volumes élémentaires.

Tous ces volumes ont surtout été étudiés en tant que volumes englobants. La détection de non intersection se base sur le même principe : trouver un axe séparant les deux objets. Le choix de l'axe est conditionné par le volume choisi. Les intersections de sphères et d'AABB sont les plus rapides, suivies de près par les k–DOP. Au contraire, le test entre OBB est plus lent, mais ces volumes offrent une meilleure approximation des objets, de même pour les enveloppes convexes. Ce compromis fait que le choix du volume dépend beaucoup du contexte et en particulier de la forme et du mouvement des objets considérés. La figure 8.4 montre un classement des volumes en fonction de la rapidité du leur test de collision et leur adéquation à englober un objet quelconque.

Les volumes englobants permettent de préciser les zones de collision [MW88, SKTK95, PML95]. Certaines structures sont basées sur des volumes empilés ou imbriqués, indépendamment des objets. C'est le cas des *R-Tree* [HKM95], des *Box-Trees* [Zac97], et des subdivisions de l'espace objet en voxels [GSF94], octrees [PML95], *Bucket-Tree* [GDO00] ou octrees sphériques [TC96]. Cependant, afin de trouver rapidement les zones de contact ou de proximité, on a souvent recours à une arborescence de volumes d'approximation (BVH). La mise en œuvre d'une telle hiérarchie peut se faire selon deux approches [BCG+96] :

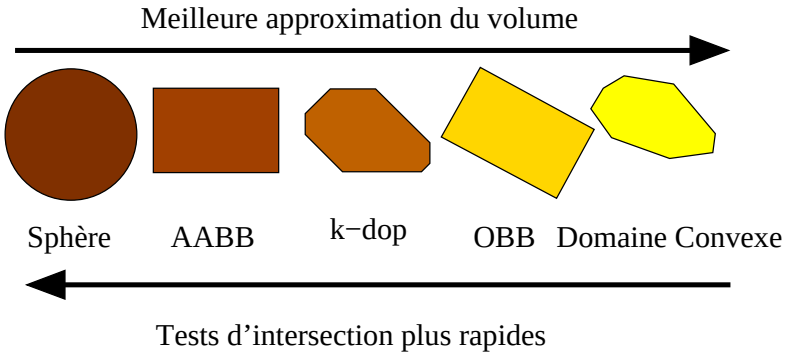

FIG. 8.4 – Comparaison entre les volumes englobants.

- approche *top-down* : c'est une approche fondée sur la construction de la hiérarchie de volumes englobants en procédant de haut en bas. En d'autres termes, l'objet entier est recouvert du volume choisi, le volume ou l'objet est ensuite subdivisé en sous-volumes, il en de même pour les sous-volumes obtenus... jusqu'à ce que l'on arrive à la primitive de bas niveau ;
- approche *bottom-up* : c'est une approche fondée sur le construction de volumes englobants en procédant de bas en haut. Dans un premier temps, de petits volumes sont distribués sur la surface de l'objet entier, la phase suivante regroupe un nombre de ces petits volumes en un volume englobant de niveau supérieur et ainsi de suite jusqu'à ce que l'objet soit complètement recouvert.

Pour détecter les collisions entre deux BVH, on commence par inspecter les volumes racines. S'il y a intersection, on descend récursivement dans la hiérarchie, selon la méthode, soit des deux objets, soit de l'objet dont le volume englobant inspecté est le plus grand. L'algorithme fournit alors un ensemble de facettes en collision probable. On trouve des hiérarchies d'OBB (OBBTrees) dans [GLM96], d'AABB dans un repère global [Hah88][LM01] ou dans le repère local de l'objet [Ber97], de sphères [PG95] [Hub96] [OD99] [RKC01], de k–DOP [KHM+98][Zac98], de Spherical-Shells [KPLM98] ou de QuOSPOs [He99]. [LM01] constatent qu'une hiérarchie à 8 fils est généralement plus efficace qu'une hiérarchie à 2 ou 4 fils, alors que [KHM+98] mesurent le contraire.

Les arborescences de volumes sont également utiles pour la recherche de bornes de distance entre objets [Qui94, JC98, LGLM00, EL01].

Ces hiérarchies sont conçues pour des polyèdres de formes complexes, souvent concaves, ou des listes de polygones sans lien topologique. Elles permettent de trouver rapidement les couples de facettes en collision ou les plus proches. Cependant, cette structure hiérarchique est plus adaptée aux corps rigides qu'aux corps déformables. En cas de déformation, il est nécessaire de recalculer les volumes en chaque nœud de l'arbre (soit $2n-1$ nœuds à inspecter pour n primitives stockées dans l'arbre).

De nombreuses optimisations, indépendante du type de volumes, mais accélérant le parcours des arborescences ont été proposées (détection à temps critique [Hub96], détection dépendante de la perception [OD99], cohérence spatiale et temporelle par *Generalized Front Tracking* [EL01].

Pour conclure sur les BVH, elles résultent d'un compromis entre :

– *la rapidité de traitement et le niveau de subdivision* : en effet, le temps de parcours et de traitement de la hiérarchie de volumes englobants peut devenir critique si l'on dépasse un certain niveau de subdivision ;
– *la mémoire de stockage* : il faut prendre en compte le coût additionnel de stockage de la

structure utilisée pour la hiérarchie ou la subdivision spatiale ;
- *la mise-à-jour des positions durant le mouvement* : cette étape peut s'avérer coûteuse en temps de calcul si elle n'est pas optimisée. En outre, certains volumes nécessitent des mises à jour en cas de simple rotation (AABB, k-DOP), et tous les volumes doivent être mis à jour pour les objets déformables ;
- *le rapport volume utile / volume englobé* : doit être proche de 1 dans le cas idéal (il n'est par exemple pas optimal d'englober une tige par une sphère, car l'espace occupé par la sphère est trop important relativement à la tige, ce qui induit potentiellement trop de tests inutiles. Un autre critère pour mesurer l'adéquation du volume englobant est de minimiser la distance de Hausdorff.

8.3.2 Stratégies exploitant le matériel graphique

Une approche originale pour la détection de collision consiste à exploiter le matériel graphique, sous réserve que tous les objets traités soient des polyèdres à face triangulaire. La partie matérielle du pipeline graphique comporte deux phases successives.

Une première famille détecte la collision en récupérant les triangles présents dans un volume englobant défini par le cône de visée (mode *feedback* [WNDS99]) [LCN99]. Cette approche est limitée par les deux seuls types de cône de visée utilisable par le matériel graphique (boîte parallélépipédique ou pyramide à base rectangulaire tronquée, éventuellement modifiable par des plans de coupe).

La second famille de solutions exploite les informations après conversion des triangles en pixels à l'écran [FS91, Zac94, BSKS99, HZLM01, KOLM02a]. Ces approches sont dites *basées sur l'image*. Leur inconvénient est la nécessité en dernière phase de l'algorithme d'inspecter l'ensemble des pixels d'un buffer à la recherche des valeurs montrant une collision. En effet, aucun accélérateur graphique ne peut répondre à une question du type : *y a-t-il un pixel qui vérifie la propriété P ?* Il faut alors choisir un compromis entre la précision (la taille d'un pixel) et le temps de recherche effectuée par logiciel, en modifiant la taille de la zone de tracé. Cependant, les approches les plus récentes [GRLM03, KP03] utilisent un système de détection d'occlusion cablée et présent sur les cartes de dernière génération pour éviter l'inspection de tous les pixels.

Enfin, une autre voie possible est de concevoir du matériel dédié à la collision [ZK03], mais ces recherches sont balbutiantes.

8.4 Accélération en mode continu

Lorsque l'algorithme de DdC est de nature continu, il serait souhaitable d'avoir un *accélérateur* cohérent, et donc continu. Les techniques à base de volumes englobants permettent d'accélérer efficacement les tests de recouvrement. Les hiérarchies de volumes englobants doivent avoir le même mouvement que l'objet.

Certaines méthodes englobent dans un même volume l'objet (ou ses volumes englobants) aux instants initiaux et finaux [ES99]. D'autres utilisent une enveloppe du volume de trajectoire [FH94] ou de nouvelles primitives 4D. Cependant, ces méthodes ne garantissent pas que la trajectoire complète de l'objet est englobée, et peuvent ainsi manquer des collisions.

Il apparaît qu'il est très facile de détecter des recouvrements en continu entre *sphères* [RKC01], et que l'on peut, dans ce cas aussi, en utilisant les mêmes fonctions d'interpolation, réduire le test de recouvrement au calcul des racines d'un polynôme de degré inférieur ou égal à celui du noyau.

Les boîtes englobantes permettent, à l'instar des sphères *accélératrices*, d'éliminer de nombreux tests élémentaires (arête/arête, point/face ou face/point) non pertinents, qui rendaient les précédentes méthodes fondées sur l'arithmétique des intervalles impraticables pour des objets polyédriques complexes. Une interpolation entre AABB peut être exploitées [MC97].

Nous avons déjà évoqué dans le tableau 8.1 le test discret le plus efficace de recouvrement entre deux OBB statiques (*théorème de l'axe séparateur* [GLM96]) :

$$\exists \vec{L} \text{ tel que } |\vec{T}.\vec{L}| > \sum_{i=1}^{3} |a_i \vec{A}_i.\vec{L}| + \sum_{i=1}^{3} |b_i \vec{B}_i.\vec{L}| \tag{8.1}$$

où $\vec{L}$ est l'axe séparateur, $\vec{T}$ est le vecteur joignant les centres de deux boîtes. La première boîte est décrite dans un système d'axe $(\vec{A_1}, \vec{A_2}, \vec{A_3})$ et par ses demi-tailles suivant ces axes (a_1, a_2, a_3). Idem pour la seconde avec $(\vec{B_1}, \vec{B_2}, \vec{B_3})$ et (b_1, b_2, b_3).

D'après le théorème de l'axe séparateur, deux OBB statiques se recouvrent si et seulement si quinze tests d'axe séparateur échouent. Les quinzes axes suffisants sont déduits des axes des OBB :

$$\vec{L} \in \{\vec{A}_i,\ \vec{B}_j,\ \vec{A}_i \wedge \vec{B}_j,\ 1 \leqslant i \leqslant 3,\ 1 \leqslant j \leqslant 3\} \tag{8.2}$$

Les termes de l'inégalité (8.1) ont une interprétation géométrique simple : le membre de gauche correspond à la distance entre les centres, suivant la direction de $\vec{L}$, et le membre de droite correspond à la somme des demi-rayons des boîtes, suivant la même direction.

Il est possible de transformer ce test discret en un test continu [RKC02a]. En effet, lorsque les boîtes englobantes se déplacent au cours d'un intervalle de temps, les deux membres de l'inégalité (8.1) sont des fonctions continues du temps. Cependant, en raison notamment des valeurs absolues intervenant dans les tests d'axe séparateur, le mouvement intermédiaire entraîne que les deux membres de l'inégalité sont des fonctions polynomiales *par morceaux*. Calculer les racines de chacun de ces morceaux de polynômes, même exactement, conduirait à un test de recouvrement très inefficace. Afin de résoudre les équations élémentaires d'une part, et de détecter continûment des recouvrements entre volumes englobants d'autre part, une approche par *l'arithmétique d'intervalles* a été proposée. L'arithmétique des intervalles est utilisée pour calculer de façon robuste les instants auxquels les primitives (points, arêtes et faces) entrent en contact, et pour dériver un test de recouvrement conservatif et *continu* entre deux OBB mobiles à partir du test de recouvrement discret.

Pour détecter des collisions en continu, il faut déterminer si deux OBB mobiles se recouvrent *durant* $[t_0, t_1]$, et pas seulement aux positions initiale et finale. Le test continu de recouvrement est constitué de deux étapes :

1. effectuer une version continue des quinze tests d'axe séparateur ;
2. si la première étape conclut que les OBB se recouvrent pour l'intervalle de temps courant, effectuer un *test de subdivision*, afin de déterminer s'il est utile de subdiviser l'intervalle de temps.

Bien que la technique précédente s'avère être bien plus pertinente que les hiérarchies de sphères, elle se révèle de moins en moins efficace à mesure que les objets se rapprochent les uns des autres et que de plus en plus de volumes englobants se recouvrent.

L'idée majeure du *test de recul hiérarchique* est de tirer parti de la *vitesse* des objets et non plus seulement de leur position, comme le font les volumes englobants. En 1994, Vanecek fait remarquer qu'un triangle qui recule relativement à un autre objet ne peut entrer en collision avec cet objet, et peut être éliminé de la liste de triangles à tester [Van94]. Cette idée est adaptée afin

d'obtenir un test de recul *hiérarchique* [RKC02b], au niveau du volume englobant. Ce test permet de détecter des situations où tous les triangles associés reculent, sans parcourir la descendance du volume englobant. Afin de réaliser ces tests de recul hiérarchiques, des informations géométriques sont ajoutées dans les volumes englobants : un nombre borné de *points caractéristiques* d'une part, qui approchent la géométrie des triangles associés et permettent d'estimer leurs vitesses et un nombre borné de vecteurs de recul d'autre part, formant un *cône de recul*, pour approcher les normales des triangles associés. Ces informations géométriques, précalculées et en nombre borné, permettront de faire un test de recul hiérarchique *en temps constant*, c'est-à-dire indépendant du nombre de triangles associés au volume englobant.

8.5 Bilan de l'accélération

Les méthodes présentées ci-dessus sont diverses et reposent sur des stratégies variées. [Zac01] propose une architecture générique permettant de sélectionner les étages du pipeline et confronte les diverses méthodes : il montre que la division de l'espace en une grille est plus efficace qu'une division hiérarchique en octrees et ne devient intéressante que pour un nombre suffisant d'objets (quelques dizaines) par rapport aux approches à base de distances inter-objets ou de plans séparateurs. Il montre également que les hiérarchies de k–DOP peuvent être plus efficaces que les hiérarchies d'OBB. Cependant, cette étude n'utilise pas certains critères pourtant complémentaires aux stratégies employées et mérite d'être complétée. En outre, les structures de données nécessaires à une détection efficace sont souvent gourmandes en mémoire et tendent à limiter la taille des modèles. Il faut alors recourir à des solutions permettant de ne charger que temporairement ces structures [WLML99].

Chapitre 9

Calcul de la réponse

Ce chapitre traite de la réponse aux collisions entre solides. La simulation de solides par modèles physiques couvre un vaste champ d'application allant des effets spéciaux cinématographiques, aux programmes de jeux vidéos en passant par toutes sortes de simulations de type éboulement rocheux, destruction d'empilements ou simulateur robotique. Les scènes de la vie quotidienne sont également composées pour la plupart d'un grand nombre d'objets solides, articulés ou non. Les méthodes d'animation par modèles physiques permettent la conception d'animation réalistes de ce type de scènes : il est en effet illusoire qu'un animateur puisse définir à la main la trajectoire des millions de grains de sables en jeu dans une simulation. Des méthodes automatiques générant des trajectoires réalistes sont nécessaires, pour simuler toutes les interactions, c'est pourquoi ce domaine a été largement exploré durant les dix dernières années.

Ce chapitre se décompose en deux parties principales. La première présente quelques notions de base sur la mécanique du solide. La seconde se compose d'un état de l'art sur la simulation de solides par modèles physiques.

9.1 Mécanique du solide

La mécanique du solide est à la base de l'implémentation d'une animation de solides par modèles physiques. Elle décrit entre autres comment un solide évolue dans l'espace au cours du temps.

9.1.1 Modélisation d'un solide

Un corps solide S_1 en mouvement, indépendemment de sa géométrie, peut être vu comme un repère local en déplacement (voir figure 9.1) , et peut donc être entièrement défini à un instant donné par :

- une position notée $\mathbf{o_i}$
- une orientation notée $\mathbf{R_i}$
- une vitesse linéaire notée $\mathbf{\dot{o}_i}$
- un vitesse angulaire notée $\boldsymbol{\omega_i}$

On note $\mathbf{q} = \left[\begin{array}{cccc} \mathbf{o_i} & \mathbf{R_i} & \mathbf{\dot{o}_i} & \boldsymbol{\omega_i} \end{array} \right]$ l'état d'un solide à un instant donné. Similairement, on note $\mathbf{\dot{q}} = \left[\begin{array}{cccc} \mathbf{\dot{o}_i} & \boldsymbol{\omega_i} & \mathbf{\ddot{o}_i} & \boldsymbol{\dot{\omega}_i} \end{array} \right]$ la dérivée de $\mathbf{q}$, où $\mathbf{\ddot{o}_i}$ et $\boldsymbol{\dot{\omega}_i}$ sont respectivement l'accélération linéaire et angulaire du solide.

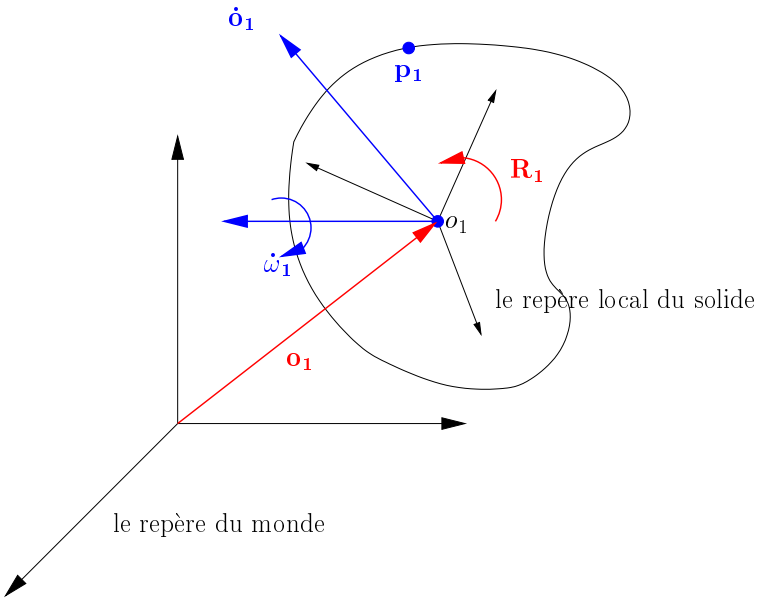

FIG. 9.1 – Représentation d'un solide : Une position $\mathbf{o_1}$ et une orientation $\mathbf{R_1}$ placent le solide dans l'espace et une vitesse $\dot{\mathbf{o}}_\mathbf{1}$ exprimée généralement dans le repère du monde ainsi qu'une vitesse angulaire $\omega_\mathbf{1}$ exprimée généralement dans le repère de l'objet. A partir de l'état $\mathbf{q}$ du solide et du vecteur $\mathbf{o_1p_1}$ il est possible de définir la vitesse et accélération d'un point p_1 du solide comme le montrent les relations de cinématique 9.6 et 9.7

9.1.2 Équations de la dynamique

Le principe fondamental de la dynamique nous donne la relation existant entre force et accélération et entre couple et accélération angulaire (9.1 et 9.2) :

$$m\ddot{\mathbf{x}} = \sum \mathbf{f_i} \tag{9.1}$$

$$\mathbf{I_M}\dot{\omega} = \sum \mathbf{c_i} \tag{9.2}$$

où m est la masse du solide considéré, $\sum \mathbf{f_i}$ la somme des forces exercées sur le solide (gravité, forces de contraintes, forces élastiques ou visqueuses ...), $\mathbf{I_M}$ la matrice d'inertie du solide et $\sum \mathbf{c_i}$ la somme des couples appliqués au solide.

La matrice d'intertie du solide a la forme suivante :

$$\mathbf{I_M} = \int_x \int_y \int_z \rho(x,y,z) \begin{bmatrix} y^2+z^2 & xy & xz \\ xy & x^2+z^2 & yz \\ xz & yz & x^2+y^2 \end{bmatrix} dxdydz$$

où $\rho(x,y,z)$ est la densité du solide en (x,y,z). Cette matrice est diagonalisable dans un repère appelé repère principal d'inertie et dont les axes sont appelés axes principaux d'inertie.

Le rôle de cette matrice est de traduire la répartition de la masse dans le solide et ainsi de favoriser certains couples. Grossièrement elle joue le même rôle pour l'accélération angulaire que la masse pour l'accélération linéaire. Par exemple il faut moins de force pour faire tourner

une barre autour de l'axe qui la traverse d'un bout à l'autre, qu'autour de l'axe qui la coupe perpendiculairement.

9.1.3 Equations de la cinématique

La cinématique du solide peut être représentée mathématiquement comme l'intégration sur le temps d'équations différentielles liant positions, vitesses et accélérations (9.3, 9.4,9.5). A partir de ces équations il est également possible de définir la vitesse et l'accélération d'un point quelconque p_1 de S_1 avec les relations de cinématique 9.6 et 9.7.

$$\dot{\mathbf{x}} = \frac{\mathbf{dx}}{\mathbf{dt}} \tag{9.3}$$

$$\ddot{\mathbf{x}} = \frac{\mathbf{d\dot{x}}}{\mathbf{dt}} \tag{9.4}$$

$$\dot{\omega} = \frac{\mathbf{d}\omega}{\mathbf{dt}} \tag{9.5}$$

$$\dot{\mathbf{p}}_\mathbf{1} = \dot{\mathbf{o}}_\mathbf{1} + \omega_\mathbf{1} \times \mathbf{o_1 p_1} \tag{9.6}$$

$$\ddot{\mathbf{p}}_\mathbf{1} = \ddot{\mathbf{o}}_\mathbf{1} + \dot{\omega}_\mathbf{1} \times \mathbf{o_1 p_1} + \omega_\mathbf{1} \times (\omega_\mathbf{1} \times \mathbf{o_1 p_1}) \tag{9.7}$$

Dans une simulation de solides par modèles physiques, les forces ne sont généralement pas appliquées aux centres de gravité des solides hors mis la force de gravité. En effet, le plus souvent, on manipule une force $\mathbf{f}$ qui est appliquée en un point $\mathbf{p_1}$. Afin de pouvoir utiliser le principe fondamental de la dynamique énoncé en 9.2, il faut au préalable ramener la force $\mathbf{f}$ au centre de gravité du solide. Il est évident que cela va induire un couple sur le solide selon la loi 9.8, qui exprime quelle force $\mathbf{f_0}$ et quel couple τ exprimés au centre de gravité o_1 du solide sont induits par une force $\mathbf{f}$ appliquée en un point $\mathbf{p_1}$ de ce même solide (voir figure 9.2).

$$\begin{bmatrix} \mathbf{f_0} \\ \tau \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{o_1 p_1} \times \mathbf{f} \end{bmatrix} \tag{9.8}$$

9.1.4 Le frottement de Coulomb

Brièvement, les lois sur le frottement de Coulomb nous disent que la force de contact $\mathbf{f}$ existant entre deux solides en contact ne peut sortir du cône de Coulomb comme le montre la figure 9.3. Cette limitation sur les composantes tangentielles de $\mathbf{f}$ s'exprime mathématiquement comme 9.9, ou $\mathbf{f} = [f_s \ f_n \ f_t]$ est exprimée dans le repère local du contact. L'impact majeur sur le mouvement d'un solide est que selon son coefficient de frottement ν et sa vitesse, il aura soit tendance à glisser sur un autre corps soit au contraire tendance à rester collé.

$$f_s^2 + f_t^2 \leq \nu^2 f_n^2 \tag{9.9}$$

9.2 La détection et modélisation de collisions

Il y a collision entre deux solides si et seulement si l'intersection de leur volume respectif n'est pas nulle. Dans le cas général de polyèdres quelconques (surtout s'ils sont non convexes) cette détection est extrêmement coûteuse si elle est faite de façon naïve. C'est pourquoi des méthodes sophistiquées ont été mises au point par des spécialistes de la géométrie algorithmique

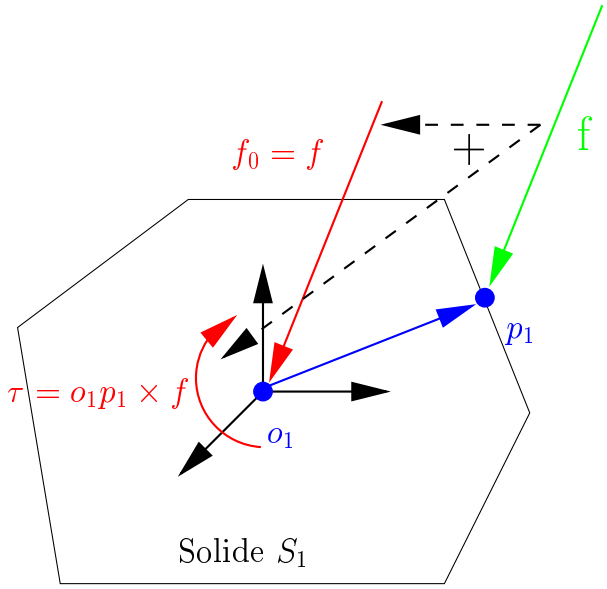

FIG. 9.2 – Appliquer une force f en un point p_1 d'un solide revient à appliquer cette même force au centre de gravité du solide ainsi qu'un couple τ qui va mettre en rotation le solide.

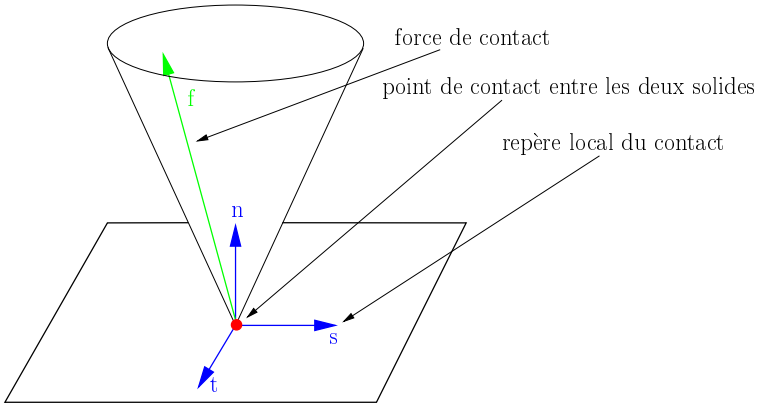

FIG. 9.3 – Cône de Coulomb : le force de contact **f** ne peut sortir du cône.

(SWIFT++, V-CLIP ...). L'intersection de deux polyèdres (creux) est un polygone. Il a été montré qu'il suffit de considérer les sommets seulement de ce polygone pour appliquer des forces par exemple, pour obtenir une simulation correcte. Pour des raisons d'efficacité, les méthodes présentées dans les sections suivantes se contenteront de faire ainsi et ne prendront donc pas en compte la surface ou volume de contact, même si cela reste possible.

La figure 9.4 montre quels sont les paramètres d'une collision à connaître. A partir de ceux-ci ainsi que des équations de la cinématique 9.6 et 9.7, il est possible de définir la notion de vitesse et d'accélération de pénétration qui sont respectivement la vitesse et l'accélération relative des points p_1 et p_2 projetés sur l'axe u.

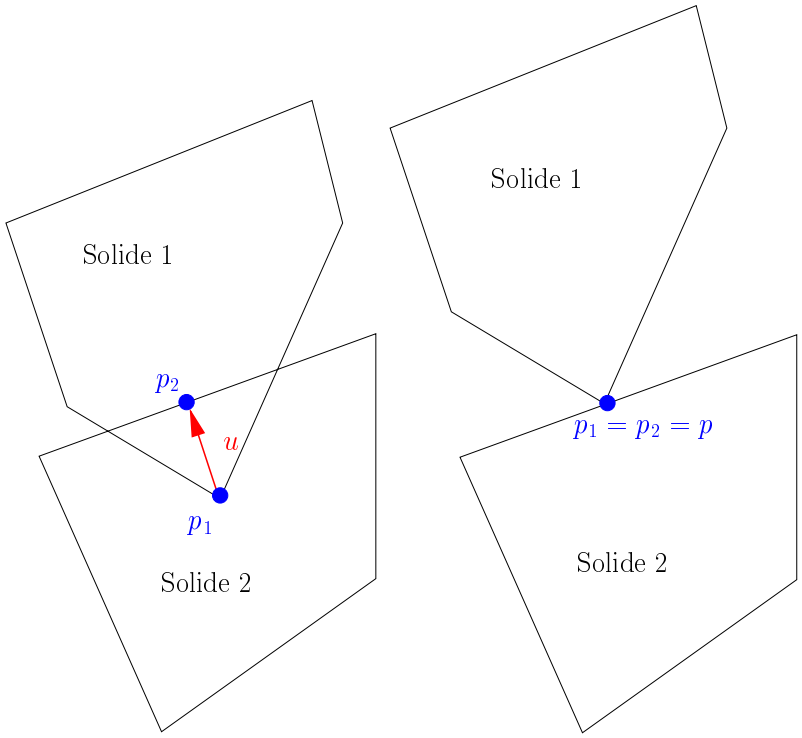

FIG. 9.4 – La modélisation d'une collision nécessite de déterminer les points p_1 et p_2 qui forment le vecteur dit d'extraction **u** donnant la plus petite translation qui une fois appliquée à l'un des deux solides en collision, va positionner ces deux solides dans un état de simple contact. Il peut exister plusieurs couples de points entre deux solides.

9.3 Principe de fonctionnement d'un modèle physique

Cette section montre brièvement à travers un schéma fonctionnel par blocs (voir figure 9.5), comment fonctionnent la plupart des modèles de simulation de solides par modèles physiques. En général l'idée est de partir d'un ensemble de solides dans un état donné, puis par intégration sur le temps des équations de la dynamique, de trouver leur nouvelles positions sans tenir compte

des collisions. Une phase de détection de collisions doit alors être effectuée. Enfin, et c'est le problème le plus ardu, une réponse aux collisions doit être calculée afin de pouvoir relancer le système dans un état valide.

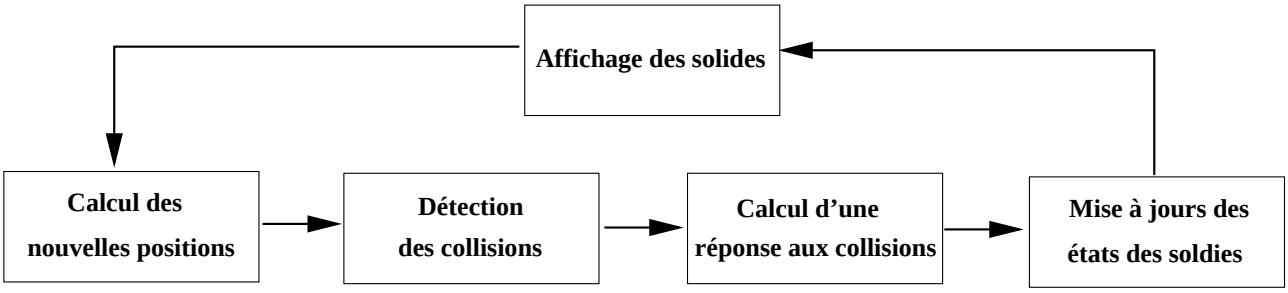

FIG. 9.5 – Schéma de fonctionnement générique d'un moteur de simulation de solides par modèles physiques.

9.4 Méthode dite de pénalité

C'est probablement l'une des premières méthodes mise au point pour la simulation de solides rigides. L'idée est en effet simple, et facile à implémenter c'est pourquoi un vaste champ de simulations l'utilise autant pour la simulation de solides rigides [MW88], [Hah88], [Mir00], [Wil87], [MZ90] ou deformable [JGT89], [Moo87],[TPB87], [TF88] que pour la simulation de tissu [BLT89].

Il s'agit de détecter toute interpénétration, et de placer un ressort virtuel de longueur à vide nulle entre tout couple de solides en collision comme le montre la figure 9.6. Ce dernier va induire une force répulsive qui va avoir tendance à faire ressortir les deux solides l'un de l'autre. En procédant ainsi, il est bien évident que l'on autorise des interpénétrations car plusieurs pas de temps éventuellement seront nécessaires pour les supprimer. Pour minimiser ce phénomène, il faut donc utiliser des pas de temps aussi petits que possible et une grande constante de rigidité pour le ressort. Dès lors que les solides se séparent (i.e. plus d'interpénétration), le ressort n'est plus nécessaire, et est donc supprimé.

Derrière cette apparente simplicité se cachent bien des problèmes. En effet, l'interpénétration peut être relativement profonde et les solides se déplacer à grande vitesse. Si des ressorts visqueux sont utilisés, la force de répulsion est directement proportionnelle à l'élongation ainsi qu'à la vitesse d'élongation comme exprimé par l'équation 9.10 où f est la norme de la force appliquée à une extréminté, k le coefficient de raideur et ν le coefficient d'amortissement (ou de viscosité), l_0 la longueur à vide du ressort (ici 0), l la longueur actuelle du ressort, $\dot{\mathbf{x}}_\mathbf{1}$ et $\dot{\mathbf{x}}_\mathbf{2}$ la vitesse des deux extrémités du ressort et u le vecteur unitaire portant la direction du ressort.

$$\mathbf{f} = \mathbf{k}(\mathbf{l} - \mathbf{l_0})\mathbf{u} + \nu(\dot{\mathbf{x}}_\mathbf{1} - \dot{\mathbf{x}}_\mathbf{2})\mathbf{u} \tag{9.10}$$

Des problèmes d'instabilité numérique peuvent alors survenir car les constantes de rigidité permettant d'éviter une plus grande interpénétration deviennent très grandes. Cela conduit à la résolution d'équations différentielles raides avec tous les problèmes que cela induit. Les schémas d'intégration explicite (euler, point milieu, Runge Kutta à l'ordre 2 ou 4) ne sont pas stables si le

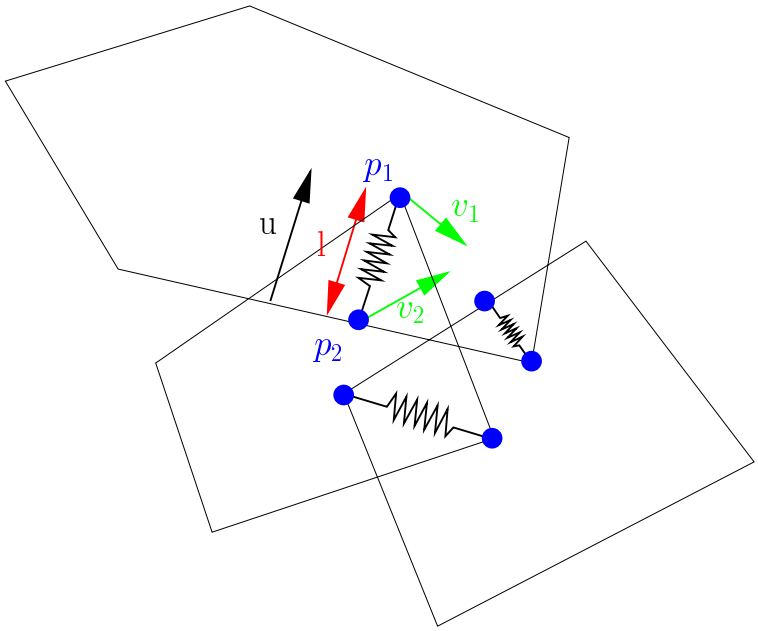

FIG. 9.6 – Illustration de la méthode de pénalité : des ressorts sont placés au niveau de chaque collision, l'élongation du ressort est l (puisque sa longueur à vide est nulle), et la vitesse d'élongation correspond à la vitesse relative des points p_1 et p_2 projetée sur l'axe $\mathbf{u}$ qui est le vecteur directeur portant le ressort, soit $\mathbf{v} = (\mathbf{v_1} - \mathbf{v_2}).\mathbf{u}$

pas de temps est plus grand que 10% de la période d'oscillation du ressort. Utiliser de tels pas de temps ralentit bien évidemment énormément la simulation et, au contraire, utiliser des ressorts trop mous entraine des phénomènes d'écrasement très indésirables, dûs à de grandes pénétrations des solides. Nous trouvons un exemple concret dans [Sch02]. Il pourrait être intéressant de tester des méthodes implicites [BW98].

À noter, que des ressorts tangentiels peuvent être ajoutés pour pouvoir prendre en compte le frottement et que l'on peut choisir une élasticité variable en compression et restitution. Ce type de méthode revient beaucoup à la mode dans le domaine de la mécanique de fracturation depuis que les performances des ordinateurs s'acroissent sans cesse. Dans ces méthodes dites d'éléments discrets, les objets sont constitués d'un assemblage de briques élémentaires (comme des sphères par exemple) liées entre elles par des liaisons de type pénalité. Les objets soumis à un stress peuvent alors se fracturer le long de ces liaisons. Cette approche locale du comportement des matériaux initialement continus est très coûteuse en temps de calcul mais bien plus appropriée (dans le cas de simulation mécanique précise) que les méthodes d'éléments finis par exemple, qui deviennent moins pertinentes lorsque les objects peuvent se fracturer (i.e. hétérogène).

9.5 Méthodes à base d'impulsion

Introduite par Hahn [Hah88], Moore et Wilhelms [MW88] et surtout Brian Mirtich [MC94], [Mir96]et [Mir98], cette classe de méthodes permet de gérer de façon unifiée collisions et contacts au repos mais aussi avec roulement, glissement ou adhérence. En effet l'état de contact au repos par exemple peut être obtenu par une série de micro-collisions qui à l'échelle macroscopique

donnent un comportement réaliste. De plus elles ont l'avantage de ne pas avoir à maintenir à jour un ensemble de contraintes comme c'est le cas dans les méthodes à base de contraintes (voire section 9.6). Enfin, utiliser des impulsions permet de passer outre les problèmes d'ambiguïté ou de non existence de solution décrit dans la section sur les méthodes à base de contraintes 9.6 ce qui conduit à une méthode beaucoup plus simple et robuste que ces dernières.

Prenons comme exemple, le paradoxe de Painlevé. Comme l'a expliqué Baraff dans [Bar91], si l'on travaille en force, avec, par exemple une méthode de pénalité (mais c'est la même chose avec une méthode à base de contrainte) et qu'au temps t_0 l'extrémité d'une barre soit dans un état de contact au repos avec le sol, donc, sans pénétration et donc sans force de contact normale comme le montre la figure 9.7. Si cependant cette barre glisse horizontalement, puisqu'il n'y a pas de force de contact normale, il n'y a pas de force de frottement tangentielle. Pourtant sous l'effet de la gravité et au pas de temps suivant la barre sera en collision avec interpénétration avec le sol. Une force de contact normale apparaît alors, ainsi qu'une force de frottement. Pour une certaine vitesse et inclinaison de la barre, celle-ci va s'arc-bouter et la pénétration va se faire encore plus grande. Les forces de contact et de frottement vont augmenter en conséquence et on part ainsi dans un cycle. Imaginons maintenant que nous augmentions la constante de raideur k du ressort de façon arbitrairement grande (nous avons le droit car les solides sont supposés rigides). En procédant ainsi la pénétration ainsi que les forces augmentent d'autant plus vite. La force de frottement s'opposant au mouvement, augmenter k permet de bloquer la barre dans son mouvement horizontal en un temps arbitrairement court, appelons le Δt. La quantité d'interpénétration est donc pendant ce temps de l'ordre de $\mathcal{O}(\Delta t^2)$ car la distance traversée verticalement par la barre dépend de façon quadratique du temps écoulé. Dans le cas limite où k tend vers l'infini, la barre est bloquée horizontalement de façon instantanée, i.e. $\Delta t = 0$. Il n'y a donc pas d'augmentation de l'interpénétration possible dans ce cas et la force de contact agit comme une impulsion sur la barre puisqu'elle modifie instantanément sa vitesse. Utiliser une impulsion dans ce cas lève donc le paradoxe.

Dans la suite nous exposerons deux méthodes permettant de calculer ces impulsions. La première est basée sur une résolution analytique d'un système d'équations et la seconde et basée sur l'intégration numérique de la vitesse de pénétration par rapport à la composante normale de l'impulsion à appliquer.

9.5.1 Méthode analytique

M. Moore et J. Wilhelms dans [MW88] ainsi que J. K. Hahn dans [Hah88] sont les premiers à mettre au point une méthode basée sur la conservation des moments angulaires et linéaires. Une impulsion pouvant être vue comme une très grande force agissant pendant un temps infinitésimal, toutes forces finies, comme la gravité par exemple, n'entrent plus en jeu car sont négligeables sur de tels intervals de temps. Ils aboutissent ainsi, pour chaque collision, au système de 15 équations à 15 inconnues 9.11 ne dépendant que des vitesses linéaires et angulaires des solides et de l'impulsion $\mathbf{r}$ entre eux (voir figure 9.8).

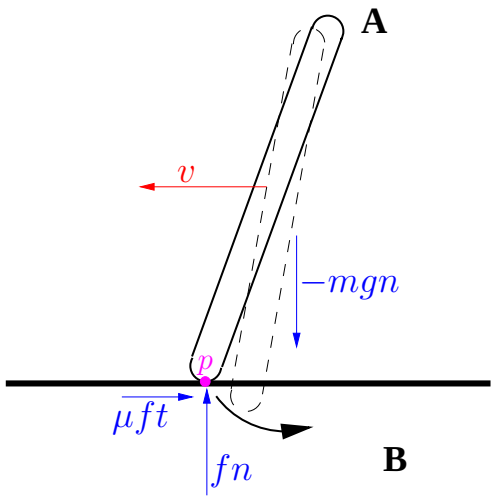

FIG. 9.7 – Illustration du paradoxe de Painlevé : pour tout $f \geq 0$ le point p de A va être accéléré dans B

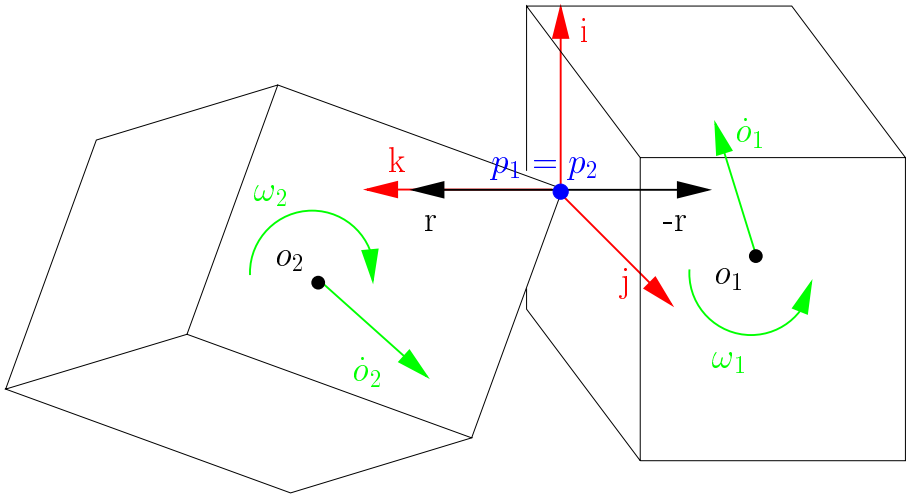

FIG. 9.8 – Soient deux solides en contact. Les vitesses au centre de gravité o_1 et o_2 de chacun des solides sont respectivement $\dot{\mathbf{o}}_\mathbf{1}$ et $\dot{\mathbf{o}}_\mathbf{2}$, et les vitesses angulaires sont $\omega_\mathbf{1}$ et $\omega_\mathbf{2}$. Au point de contact $p_1 = p_2$ le calcul de l'impulsion $\mathbf{r}$ sastisfaisant le système d'équation 9.11 va permettre de mettre à jour les vitesses linéaires et angulaires des deux solides considérés.

$$\left\{\begin{array}{rcl} m_1\dot{\mathbf{o}}_{\mathbf{1_{corrige}}} &=& m_1\dot{\mathbf{o}}_{\mathbf{1}} + \mathbf{r} \\ m_2\dot{\mathbf{o}}_{\mathbf{2_{corrige}}} &=& m_2\dot{\mathbf{o}}_{\mathbf{2}} + \mathbf{r} \\ I_1\omega_{\mathbf{1_{corrige}}} &=& \mathbf{I_1}\omega_{\mathbf{1}} + \mathbf{o_1p_1} \times \mathbf{r} \\ I_2\omega_{\mathbf{2_{corrige}}} &=& \mathbf{I_2}\omega_{\mathbf{2}} - \mathbf{o_2p_2} \times \mathbf{r} \\ \mathbf{r.i} &=& \mathbf{0} \\ \mathbf{r.j} &=& \mathbf{0} \\ (\dot{\mathbf{o}}_{\mathbf{2_{corrige}}} + \omega_{\mathbf{2_{corrige}}} \times \mathbf{o_2p_2} - \dot{\mathbf{o}}_{\mathbf{1_{corrige}}} - \omega_{\mathbf{1_{corrige}}} \times \mathbf{o_1p_1}).\mathbf{k} &=& \mathbf{0} \end{array}\right. \quad (9.11)$$

Ce système est ensuite résolu par une méthode de type Gauss-Jordan ou décomposition LU. La gestion du rebond se fait une fois le système résolu, en remplaçant r par $(1+\epsilon)\mathbf{r}$, i.e. en utilisant le modèle de poisson et en recalculant les $\dot{\mathbf{o}}_{\mathbf{i_{corrige}}}$ et $\omega_{\mathbf{i_{corrige}}}$.

La gestion du frottement se fait également en deux temps, mais nécessite cette fois deux résolutions du système 9.11. Une première résolution permet de calculer $\mathbf{r}$. Si $\mathbf{r}$ est à l'intérieur du cône de frottement, alors le calcul est terminé, sinon on résoud le système une seconde fois en posant des contraintes sur les valeurs tangentielles de $\mathbf{r}$ qui vont garantir que ce vecteur soit sur le cône de frottement. Ceci se fait simplement en remplacant dans le système les deux lignes $\mathbf{r.i} = \mathbf{0}$ et $\mathbf{r.j} = \mathbf{0}$ par 9.12 ou les deux coefficients α et β on été calculés de façon appropriée afin de garantir la contrainte de position sur le cône.

$$\begin{array}{rcl} \mathbf{r.i} &=& \alpha\mathbf{r.k} \\ \mathbf{r.j} &=& \beta\mathbf{r.k} \end{array} \quad (9.12)$$

Le problème avec cette classe de méthodes est qu'elle traite les collisions une par une et qu'elle n'est valide que sur deux solides en contact, mais pas en interpénétration, ce qui implique la recherche de l'instant précis ou deux solides entrent en contact (voir figure 9.9). Etant donné que l'intégration du temps se fait de façon discrète, à chaque pas de temps il faut détecter toutes les collisions et remonter dans le temps afin de trouver ou c'est formé le premier contact. La résolution du système 9.11 est ensuite appliquée à celui-ci, mais il faut maintenant trouver l'instant du deuxième contact et ainsi de suite. Le problème est que cela engendre de très nombreux retours en arrière et de fréquents arrêts de la simulation pour chaque résolution de système. Etant donné que le nombres de collisions augmente très vite en fonction du nombre de solides en jeu dans la simulation ce type de méthode devient très vite impraticable, a fortiori pour une application temps réel.

Une alternative à cette méthode de « Retroactive detection »(RD) a cependant été proposée par Mirtich. Nommée « Conservative Advancement »(CA) elle évite d'effectuer des calculs inutiles induits par tous ces retours arrière dans le temps. Une borne inférieure sur le prochain instant d'impact est fixée. Le système entier peut alors être intégré jusqu'à ce moment tout en évitant des collisions multiples. Le travail perdu est alors limité car l'instant de collision exact n'est plus calculé une fois que la simulation est allée trop loin. Bien sûr cela ne fait que repousser le problème car pour des scènes ou de nombreux solides sont en grande proximité, comme par exemple dans des empilements, le pas de temps de sécurité à choisir devient alors extrêmement petit et les temps de calculs augmentent alors en fonction.

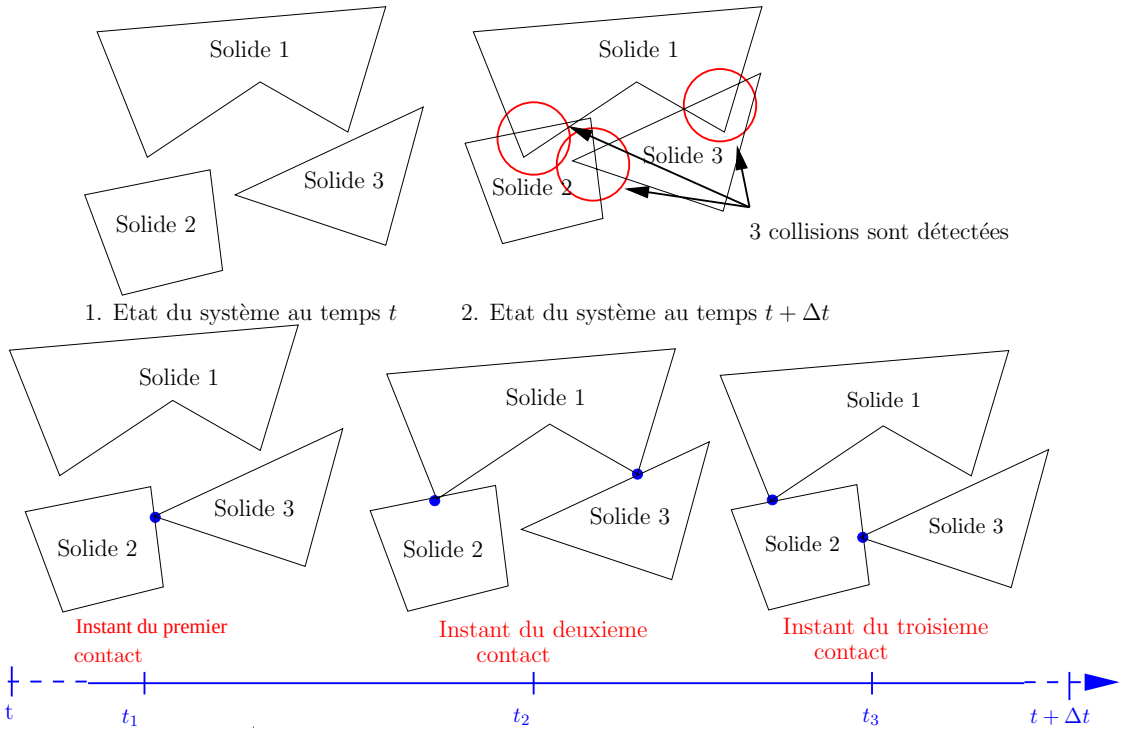

FIG. 9.9 – Au temps $t+\Delta t$ plusieurs collisions sont détectées. Il faut donc remonter dans le temps, au moment de la première collision, afin d'appliquer la résolution du système. La simulation est ensuite relancée à partir de cet état.

9.5.2 Par intégration numérique

L'idée développée par Mirtich dans [MC94], [Mir96] et [Mir98] est de considérer la collision comme ayant une durée infinitésimale et de la décomposer en une phase de compression ainsi qu'une phase de restitution, la limite entre les deux étant le point de compression maximum, c'est-à-dire le moment ou la vitesse de pénétration est annulée. Pour déterminer le point de compression maximum Mirtich propose d'exprimer la dérivée de la vitesse de pénétration u' en fonction du coefficient de frottement μ et de la vitesse de pénétration u. Cela nous mène à l'équation différentielle non linéaire 9.13.

$$\begin{bmatrix} u'_x \\ u'_y \\ u'_z \end{bmatrix} = M \begin{bmatrix} -\mu \frac{u_x}{\sqrt{u_x^2+u_y^2}} \\ -\mu \frac{u_y}{\sqrt{u_x^2+u_y^2}} \\ 1 \end{bmatrix} \tag{9.13}$$

Par intégration numérique par rapport à la composante normale de l'impulsion à appliquer il est possible de suivre l'évolution de u durant la collision et ainsi l'impulsion appliquée puisqu'il existe une relation linéaire entre une variation de u et celle-ci. Lorsque $u = 0$ le point de compression maximum est atteint et le problème est résolu. Si par contre lors de l'intégration la composante tangentielle de u s'annule, ceci signifie que l'on passe dans une phase d'adhérence puisque les deux solides considérés ne glissent plus l'un sur l'autre. Dans ce cas, la direction de l'impulsion n'est plus connue à priori (puisqu'elle n'est plus sur le cône de Coulomb). Il est alors nécessaire dans un premier temps de vérifier si l'impulsion correspondante se situe bien à l'intérieur du cône de frottement. Si ce n'est pas le cas, la force de frottement n'est pas suffisante pour maintenir l'adhérence et l'on repart en glissement et l'intégration continue comme auparavant, sinon on arrête l'intégration et on maintient les valeurs calculée pour le reste de la phase de collision.

9.6 Méthodes à base de contraintes

Introduite par Lötstedt dans [Löt81] et développées par Baraff dans toute une série d'articles [Bar94], [Bar89] et [Bar90], l'idée est ici de trouver un ensemble de forces à appliquer au niveau des points de contact afin de prévenir les interpénétrations. Ces forces n'effectuent aucun travail sur les objets et traduisent simplement une contrainte de non pénétration. Par exemple une sphère qui roule sur un plan doit avoir son centre de masse décrivant une ligne parallèle au plan en question située hauteur égale au rayon de la sphère.

Bien évidemment les forces appliquées doivent êtres réciproques, c'est-à-dire que pour un point de collision i entre un solide A et B, si l'on choisit par convention d'appliquer la force f_i appliquée sur A, la force $-f_i$ doit être appliquée sur B. Intuitivement f_i doit être positive, c'est-à-dire répulsive car elle doit éloigner les deux solides l'un de l'autre et non les attirer l'un vers l'autre. De plus, pour empêcher une interpénétration il faut que l'accélération de pénétration au contact i après application de la force f_i soit nulle. De plus les contraintes à appliquer sont dites unilatérales car bien que l'interpénétration soit interdite, le décollement est autorisé. Cela nous mène à la deuxième condition à satisfaire. En effet après application des f_i on autorise que les a_i soient positives, c'est-à-dire que les solides s'éloignent l'un de l'autre. Bien sur, dans ce cas, la force f_i correspondante doit être nulle. Ces conditions sont résumées figure 9.10.

Lödstedt dans [Löt81] a montré que l'on peut formuler ce problème à l'aide d'un LCP (Linear Complementarity Problem) de la forme suivante :

$$\begin{cases} \mathbf{a} & = & \mathbf{Af} + \mathbf{b} \\ \mathbf{a} & \geq & \mathbf{0} \\ \mathbf{f} & \geq & \mathbf{0} \\ f_i a_i & = & 0 \end{cases} \tag{9.14}$$

Ce LCP peut également s'écrire sous la forme 9.15, en remarquant que puisque $a_i \geq 0$ et $f_i \geq 0$ on a :

$$\sum_{i=1}^{n} f_i a_i = 0 \Leftrightarrow \mathbf{f^T a} = \mathbf{0}$$

$$\begin{cases} \mathbf{Af} + \mathbf{b} & \geq & \mathbf{0} \\ \mathbf{f} & \geq & \mathbf{0} \\ \mathbf{f^T}(\mathbf{Af} + \mathbf{b}) & = & \mathbf{0} \end{cases} \tag{9.15}$$

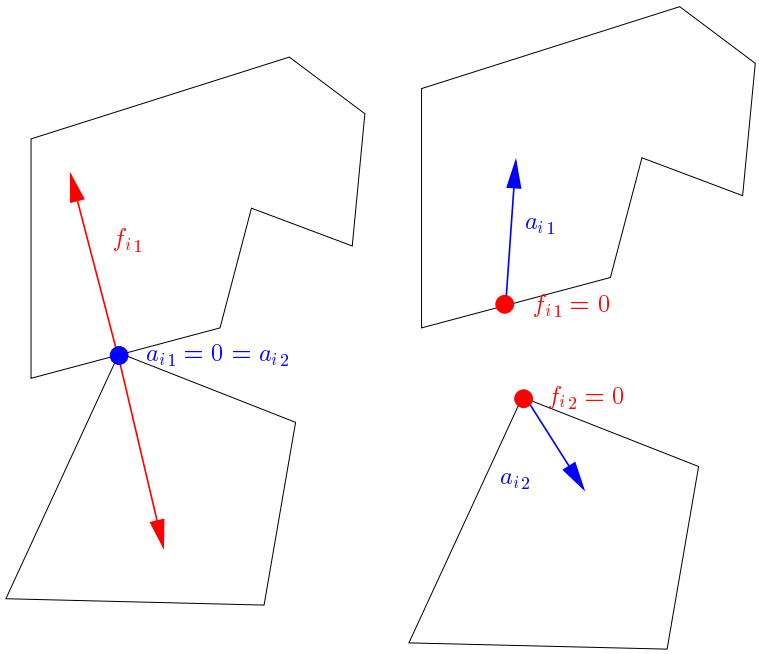

FIG. 9.10 – À gauche les deux solides sont en contact et $a_i = 0$, $f_i > 0$, à droite les deux solides sont en décollement et aucune force n'est appliquée aux points de contacts précédents, par contre leur accélération relative n'est pas nulle.

où a est un vecteur contenant tous les a_i et f un vecteur contenant les f_i. A est une matrice modélisant la géométrie des contacts ainsi que les masses et inerties des solides. Les trois dernières conditions traduisent le fait que soit les solides s'éloignent (i.e. une accélération de pénétration positive existe) et aucune force n'agit entre eux soit les solides sont en contact et une force positive maintient cet état de contact et empêche l'interpénétration mais l'accélération de pénétration est nulle.

Il s'agit maintenant de résoudre ce LCP. Cottle et al dans [RKCS92], Chan et Pang dans [CP82] ainsi que Murty dans [Mur97], discutent des méthodes de résolutions. Baraff propose une version modifiée de l'algorithme de Dantzig dans [Bar94].

L'avantage de procéder ainsi et que tous les contacts sont traités en même temps, dans un seul et même système, ce qui est beaucoup plus rapide que de traiter les contacts un à un. L'inconvénient par contre, est que ces méthodes ne peuvent traiter les collisions avec des vitesses de pénétrations mais seulement les contacts avec vitesses de pénétration nulles. Les collisions doivent donc être traitées avec une autre méthode, comme par exemple une méthode à base d'impulsion présentée section 9.5.1.

Bien qu'une solution existe toujours pour le cas sans frottement, ce n'est plus du tout le cas lorsque l'on introduit celui-ci. Deux cas problématiques peuvent alors apparaître. Un cas est dit ambigu ou indéterminé car deux ou plusieurs solutions sont possibles ; le second est dit inconsistant car aucune solution ne satisfait le LCP. De nombreux articles traitent de ces problèmes [SS98], [ST97], [Ste00], [ST96] ou [AP97]. Baraff par exemple dans [Bar91] montre de plus que le problème de détermination des forces de contact avec frottement est NP dur. Il apporte également une solution en utilisant des forces impulsives (voir 9.5.1).

9.7 Méthode dite *timewarp*

De bons algorithmes existent donc pour la simulation de solides par modèles physique. Mais dès lors que le nombre de corps s'accroît trop, le nombre de collisions croissant par pas de temps rend beaucoup de ces méthodes inutilisables. C'est pourquoi Mirtich dans [Mir00] propose une alternative baptisée *timewrap rigid body simulation*. Pour lui le problème n'est pas dans les composants de boucle de simulation comme par exemple les méthodes de réponses ou de détection des collisions mais c'est la boucle elle-même qui est le facteur limitant. En effet, elle impose généralement une synchronisation entres les corps qui n'est pas forcément nécessaire ; c'est le cas dans les méthodes de CA ou RD. Il reprend alors le paradigme de Jefferson mis au point originellement dans le domaine des systèmes et réseaux informatiques. L'idée est une désynchronisation de la simulation en exploitant les caractères discrets des simulations de solides rigides. Il a en effet remarqué qu'il n'est pas forcément nécessaire de maintenir une horloge de simulation globale à tous les corps. Les solides sont en effet intégrés dans des groupes de solides. Deux groupes n'étant pas en interaction, il n'est pas nécessaire de regarder les collisions entre leur constituant. De plus chaque groupe possède un temps local virtuel différent qui leur permet d'avancer dans la simulation de façon indépendante par des méthodes de type CA ou RD par exemple. Chaque groupe étant réduit, les performances ne deviennent pas catastrophiques avec ces méthodes classiques. Un système de messages, anti-messages et queues de messages permet de gérer les interactions entre les différents groupes et si nécessaire des les fusionner ou de les diviser. Un temps global virtuel égal au minimum des temps locaux virtuels est défini afin de pouvoir faire interagir les différents groupes. Des retours en arrière sont bien sûr possibles lorsque deux groupes entrent en interaction mais cela n'influe pas sur tous les solides en jeu dans la simulation. En procédant ainsi Mirtich peut utiliser des pas de temps de deux à seize fois plus grands qu'en utilisant des méthodes de type CA ou RD.

Bien évidemment, cette méthode trouve également ses limites lorsque les solides simulés sont en grande proximité comme dans les empilements ou amoncellements par exemple. En effet dans ce cas, très peu de groupes existent, voire un seul uniquement. Dans ce cas, on retombe sur des performances des méthodes de RD et CA, voire inférieures à cause de tout le système de messages sous jacent. Par contre, dans le cas de simulations ou les solides sont relativement éparpillés dans l'espace, comme dans des éboulements peu denses, les groupes sont plus nombreux et l'on voit apparaître les avantages de cette méthode ; de plus les groupes étant indépendants, elle ouvre la voie à la parallélisation des simulations de solides rigides, les envois de messages pouvant

parfaitement bien se faire via un réseau.

9.8 Méthodes à base d'optimisation : OBA

Ce paradigme d'animation basé sur l'optimisation a été introduit par Milenkovic et Schmidl. Milenkovic en a posé les bases dans l'article [Mil96] en 1996. Sa technique lui a permis d'animer 1000 sphères dans une sorte de sablier en utilisant uniquement de la physique à l'ordre zéro, c'est-à-dire qu'il n'y pas de notion ni de vitesse ni d'accélération. Le frottement et le rebond ne sont également pas modélisés. Les contraintes de non pénétration sont atteintes en utilisant une méthode empruntée aux algorithmes de compaction d'objets dans un volume déterminé (utilisés dans l'industrie du textile par exemple pour minimiser les chutes de tissus), basée sur le concept de programmation linéaire, dont la fonction objectif amène les sphères aussi proche que possible de là où elles veulent être, en minimisant une certaine énergie potentielle. Les solides ne pouvant que se translater, sans aucune rotation, cette technique reste très limitée (à des sphères notamment). De plus la gravité doit être simulée artificiellement et le réalisme est assez faible car les sphères ont tendance à s'aligner selon des directions préférentielles, en forme de diamant. Ce problème est dû du choix de l'algorithme du simplexe qui cherche la solution optimale aux extrémités seulement des régions de faisabilités.

Fort de cette nouvelle approche, Schmidl proposa avec succès de la généraliser à la physique d'ordre supérieur, proposant ainsi une alternative aux algorithmes type RD et CA, nommée OBA. Dans [MS01] puis dans sa thèse [Sch02], il propose d'utiliser de grand pas de temps pour pouvoir avoir une simulation plus rapide, mais qui n'est que visuellement réaliste même si le réalisme physique n'est pas total, ce qui est tout à fait acceptable pour toutes les applications non critiques. Des pas de temps égaux aux intervalles entre deux images sont fixés (pas de sous pas de temps donc). La correction des états des solides se fait en trois temps. Premièrement la trajectoire des solides est intégrée sur le temps. Pour chaque solide nouvellement en collision un algorithme de programmation quadratique tente à partir de leur position initiale avant intégration des les rapprocher autant que possible tout en évitant l'interpénétration. Cette étape est rendue possible grâce à l'utilisation de la notion de plan séparateur, qui n'existe entre deux solides, que si ceux-ci ne sont pas en collision et qui deviennent unique si les deux solides sont en contact. Tous les contacts et collisions sont donc synchronisés à la fin du pas de temps, bien que le mouvement des solides soit désynchronisé puisque chacun aura parcouru un intervalle de temps différent jusqu'à entrer en contact avec un autre corps. Dans un second temps la mise à jour des moments se fait par impulsions en résolvant également une série de programmes quadratiques dont la fonction objectif est la minimisation de l'énergie cinétique. Cela évite d'une part d'ajouter de l'énergie au système et d'autre part stabilise la simulation en dissipant autant que possible d'énergie. Une approximation du cône de Coulomb en pyramide cônique à huit faces est utilisée pour la gestion du frottement afin de pouvoir poser des contraintes linéaires entre les composantes tangentielles et normales des impulsions à calculer. Le modèle de Newton est utilisé pour le rebond. La troisième étape consiste à calculer les forces de contact afin d'éviter que les solides ne rentrent en interpénétration. Un procédé similaire à la mise à jour des moments est utilisé.

On remarque que cette méthode gérant les contraintes unilatérales peut également être étendue à la gestion des contraintes bilatérales, i.e. à la gestion des solides articulés.

Chapitre 10

Aspects pratiques : les solutions complètes

Suivant les applications, diverses solutions, commerciales ou disponibles dans le domaine public, ont été proposées. Nous les présentons dans ce chapitre.

10.1 Les pipelines de détection de collision

Toutes les stratégies que nous avons exposées précédemment sont très différentes. Il apparaît donc très intéressant d'utiliser en cascade plusieurs de ces méthodes [PG95]. On trouve au départ du pipeline la phase de détection de proximité suivie de la phase de détection approximative. Dans tout le pipeline, les méthodes très rapides, mais imprécises, sont suivies par les méthodes les plus précises mais les plus coûteuses en temps. La dernière étape du pipeline est constituée par la détection exacte la plus appropriée au contexte.

De nombreuses solutions ont été proposées et sont souvent disponibles sur le WEB : V-Collide [LMCG96], Q-Collide [CW96], V-Clip [Mir97] ou H-Collide [GLGT99]. La figure 10.1 compare les approches utilisées par Q-Collide et V-Collide.

Le tableau 10.1 résume la plupart des bibliothèques[1].

Outils	Méthodes	Entrées	Sorties	Catégorie
RAPID	OBB-Trees	Polygones	Couple de Faces	2Body
EPA	GJK	Polyèdres convexes	Distance+Interpénétration	2body
Solid	ABBTrees	Polygones	Point de contact	Nbody
ICollide	Voronoï	Polyèdres convexes	Distance	Nbody
V-Clip	ICollide	Polyèdres convexes	Distance	NBody
Swift	Voronoï+Multilevel	Polyèdres convexes	Distance	Nbody
Swift++	Voronoï+Multilevel	Polyèdres	Distance	Nbody
VCollide	S&P + Rapid	Polygones	Couple de faces	Nbody
Q-Collide	S&P + séparation + GJK	Polyèdres convexes	Couple de Faces	NBody
DEEP	Carte de Gauss	Polyèdres convexes	Pénétration	2Body
Quick-CD	K-Dops trees	Polygones	Coupes de triangles	2Body
VPS	Voxels	Points	Pénétration	2Body
Hcollide	Voxels+OBBTrees	Polyhèdres	Pénétration	2Body
Contact	OBB-Trees en continu	Polygones	Point de contact	2Body
ColDet	OBB-Trees	Polygones	Collision exacte	2Body

TAB. 10.1 – Libraries de DdC existantes.

[1] CT = Cohérence Temporelle, S&P = Sweep and Prune.

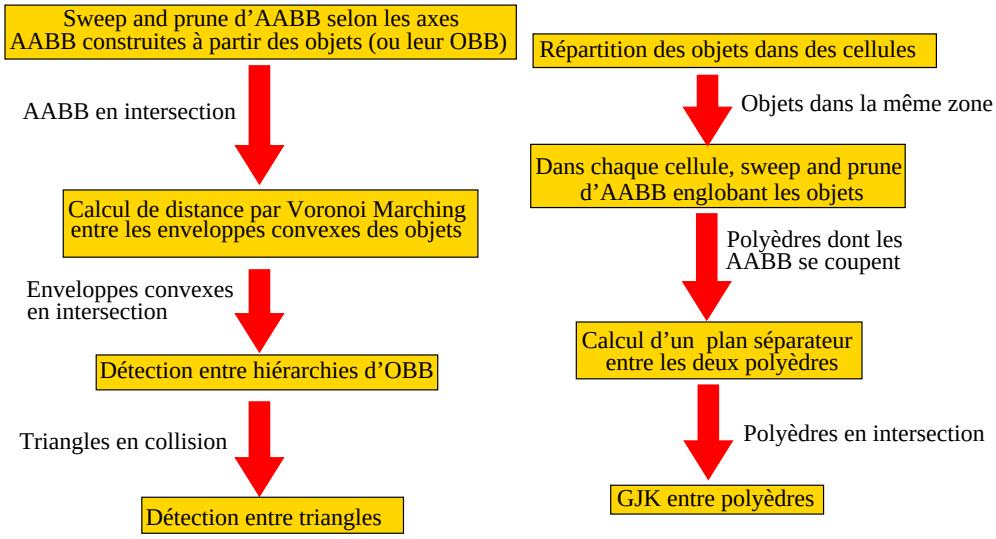

FIG. 10.1 – Étapes utilisées par deux bibliothèques standard.

10.2 Détection de collisions pour le retour d'effort

Plusieurs bibliothèques ont été proposées pour gérer des systèmes à retour d'effort. Ces bibliothèques gèrent principalement les collisions entre l'objet manipulé et l'environnement virtuel, et en déduisent les efforts à retourner. Ces bibliothèques sont étudiées pour calculer rapidement les collisions, à la fréquence nécessaire au dispositif. Aucune bibliothèque ne gère de simulation basée sur la physique, cela est dévolu à l'application.

La bibliothèque Ghost fournie avec le dispositif PHANToM de Sensable gère une liste limitée d'objets avec leur boîte englobante. La méthode *H- Collide* [GLGT99] modélise un environnement statique par une grille régulière de l'espace et l'utilisation d'arbres OBB dans chaque cellule. [DMC02] utilise une décomposition des objets en sphères qui sont réparties dans une grille régulière de l'espace. [HKM95] découpent l'espace en un maillage tétraèdrique conforme. [RKK97] se basent sur des classiques hiérarchies de sphères. Enfin, la méthode *Voxmap Pointshell* (VPS) de [MPT99] se base sur un environnement statique décrit en géométrie discrète (voxels) et un échantillonnage surfacique de l'objet manipulé.

On constate donc que l'on trouve les mêmes stratégies de détection de collisions que nous avons présentées dans l'état de l'art précédent. Cependant, on voit que les environnements sont souvent statiques et les méthodes de type *2-body*. Certaines tentatives ont essayé d'appliquer des méthodes *N-body* pour des objets polyédriques convexes [GME+00] ou quelconques [KOLM02b] [HBS99], où toutefois le nombre de corps simulés est très réduit (une dizaine). En effet, la détection de collisions dédiée au retour d'effort se doit de respecter des contraintes fréquentielles strictes. Elle se doit donc d'être très efficace et à temps garanti. Si l'environnement est dynamique et complexe, la détection de collision est généralement trop lente, aussi on a souvent recours à une représentation intermédiaire ou *buffer-model* [AKO95, Bal99], offrant souvent une détection de collisions entre un nombre réduits d'objets ou primitives d'objets extraits de la simulation.

Bibliographie

[AGHP+00] P. Agarwal, L. Guibas, S. Har-Peled, A. Rabinovitch, and M. Sharir. Computing the penetration depth of two convex polytopes. *Nordic Journal on Computing*, 7 :227–240, 2000.

[AKO95] Y. Adachi, T. Kumano, and K. Ogino. Intermediate representation for stiff virtual objects. In *IEEE VRAIS'95*, pages 203–210, Mars 1995.

[AP97] M. Anitescu and F. Potra. Formulating dynamic multi-rigid-body contact problems with friction assolvable linear complementarity problems, 1997.

[Bal99] R. Balaniuk. Using fast local modelling to buffer haptic data. In *4th PHANToM Users Group Workshop*, Boston, Octobre 1999.

[Bar89] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics*, 23 :223–232, 1989.

[Bar90] D. Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *Computer Graphics (Proceedings of ACM SIGGRAPH 90)*, 24(4) :19–28, Août 1990.

[Bar91] David Baraff. Coping with friction for non-penetrating rigid body simulation. *Computer Graphics*, 25 :31–40, 1991.

[Bar94] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94*, pages 23–34. ACM SIGGRAPH, ACM Press, 1994.

[BCG+96] G. Barequet, B. Chazelle, L. Guilbas, J. Mitchell, and A. Tal. Boxtree : A hierarchical representation for surfaces in 3d. *Computer Graphics Forum (Proceedings of EUROGRAPHICS'96)*, 15(3) :387–396, Août 1996.

[Ber97] G. Van den Bergen. Efficient collision detection of complex deformable models using aabb trees. *Journal of Graphics Tools*, 2(4) :1–14, 1997.

[Ber99] G. Van den Bergen. A fast and robust gjk implementation for collision detection of convex objects. *Journal of Graphics Tools*, 4(2) :7–25, 1999.

[Ber01] G. Van den Bergen. Proximity queries and penetration depth computation on 3d game objects. In *Game Developper Conference*, 2001.

[BF79] J.L. Bentley and J.H. Friedman. Data structures for range searching. *ACM Computing Surveys*, 11(4) :398–409, Décembre 1979.

[BLT89] N.M. Thalmann B. Lafleur and D. Thalmann. Cloth animation with self collision detection. *Modeling In Computer Graphics*, 1989.

[Boy79] J. W. Boyse. Interference detection among solids and surfaces. *Communications ACM*, 22(1) :3–9, Janvier 1979.

[BSKS99] G. Baciu, W. Sai-Keung, and H. Sun. Recode : an image-based collision detection algorithm. *Journal of Visualization and Computer Animation*, 10 :181–192, 1999.

[BT95] S. Bandi and D. Thalmann. An adaptative spacial subdivision of the object space for fast collision detection of animated rigid bodies. *Computer Graphics Forum (Proceedings of EUROGRAPHICS'95)*, 14(3) :259–270, Août 1995.

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Computer Graphics (Proceedings of SIGGRAPH '98)*, pages 43–54. ACM SIGGRAPH, Juillet 1998.

[BWK03] D. Baraff, A. Witkin, and M. Kass. Untangling cloth. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003)*, 22(3), Juillet 2003.

[Cam90] S. Cameron. Collision detection by four-dimensional intersection testing. *IEEE Transactions on Robotics and Automations*, 6(3) :291–302, Juin 1990.

[Cam97] S. Cameron. Enhancing gjk : Computing minimum and penetration distance between convex polyhedra. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3112–3117, 1997.

[Can86] J. Canny. Collision detection for moving polyhedra. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2) :200–209, Mars 1986.

[CC86] S.A. Cameron and R.K. Culley. Determining the minimum translational distance between two convex polyhedra. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 591–597, 1986.

[CK86] R.K. Culley and K.G. Kempf. A collision detection algorithm based on velocity and distance bounds. In *IEEE International Conference on Robotics and Automations (ICRA)*, pages 1064–1069, 1986.

[CLMP95] J.D. Cohen, M.C. Lin, D. Manocha, and M.K. Panamgi. I-collide : An interactive and exact collision detection system for large-scale environments. In *ACM Interactive 3D Graphics Conference*, pages 189–196, 1995.

[CP82] D. Chan and J.S. Pang. Iterative methods for variational and complementarity problems. In *Mathematical Programming*, volume 24, pages 284–313, 1982.

[CW96] K. Chung and W. Wang. Quick elimination of non-interference polytopes in virtual environment. In *Third European Workshop on Virtual Environments*, Monté-Carlo, Février 1996.

[Dan92] M. Daniel. A curve intersection algorithm with processing of singular cases : introduction of a clipping technique. In T. Lyche and L. S. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design II*, pages 161–170, 1992.

[DCG94] M. Desbrun and M.P. Cani-Gascuel. Highly deformable material for animation and collision processing. In *EUROGRAPHICS Workshop on Computer Animation and Simulation (EGCAS)*, Oslo, 1994.

[DHKS93] D. Dobkin, J. Hershberger, D. Kirkpatrick, and S. Suri. Computing the intersection-depth of polyhedra. *Algorithmica*, 9 :518–533, 1993.

[DK90] D.P. Dobkin and D.G. Kirkpatrick. Determining the separation of preprocessed polyhedra : a unified approach. In *17th International Conference on Automata Language Programming*, pages 400–413, 1990.

[DMC02] J Davanne, P. Meseure, and Chaillou C. Stable haptic interaction in a dynamic virtual environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Octobre 2002.

[Duf92] T. Duff. Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 2(26) :131–138, Juillet 1992.

[Dwo94] P.J. Dworkin. Efficient collision detection for real-time simulated environments. Master's thesis, Massachusetts Intitute of Technology, 1994.

[DZ93] P. Dworkin and D. Zelter. A new model for efficient dynamic simulation. In *EUROGRAPHICS Workshop on Computer Animation and Simulation (EGCAS)*, pages 135–147, Barcelone, Septembre 1993.

[EHK+00] B. Eberhardt, J. Hahn, R. Klein, W. Strasser, and A. Weber. Dynamic implicit surfaces for fast proximity queries in physically based modeling. In *Graphisch-Interaktive Systeme (WSI/GRIS)*, 2000.

[EL00] S. Ehmann and M.C. Lin. Accelerated proximity queries between convex polyhedra by multi-level voronoi marching. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2000.

[EL01] S. Ehmann and M.C. Lin. Accurate and fast proximity queries between polyhedra using convex surface decomposition. *Computer Graphics Forum (Proceedings of EUROGRAPHICS'01)*, 20(3), Septembre 2001.

[ES99] J. Eckstein and E. Shömer. Dynamic collision detection in virtual reality applications. In *WSCG'99 Conference*, pages 71–78, Plzen, Février 1999.

[FDFH90] J.D. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics – Principles and Practice.* Addison Wesley, seconde édition, 1990.

[FH94] A. Foisy and V. Hayward. A safe swept volume method for collision detection. In *6th International Symposium of Robotics Research*, pages 62–68, Cambridge, 1994.

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

[FS91] M.C. Forgue and M. Shinya. Real-time impact dynamics simulation. In *Compugraphics*, pages 427–435, 1991.

[GDO00] F. Ganovelli, J. Dingliana, and C. O'Sullivan. Buckettree : Improving collision detection between deformable objects. In *Spring Conference on Computer Graphics*, pages 156–163, 2000.

[Gei00] B. Geiger. Real-time collision detection and response for complex environment. In *Computer Graphics International Conference*, Genève, Juin 2000.

[GH89] E.G. Gilbert and S.M. Hong. A new algorithm for detecting the collision of moving objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1989.

[GHZ99] L.J. Guibas, D. Hsu, and L. Zhang. H-walk : Hierarchical distance computation for moving convex bodies. In *15th Annual Symposium on Computational Geometry*, 1999.

[GJK88] E.G. Gilbert, D.W. Johnson, and S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of robotics and Automation*, 4(2) :193–203, Avril 1988.

[GLGT99] A. Gregory, M.C. Lin, S. Gottshalk, and R. Taylor. A framework for fast and accurate collision detection for haptic interaction. In *IEEE Virtual Reality Conference*, 1999.

[GLM96] S. Gottschalk, M.C. Lin, and D. Manocha. Obbtree : a hierarchical structure for rapid interference detection. In *SIGGRAPH 96 Conference Proceedings*, Computer Graphics annual conference series, pages 171–180, New Orleans, Août 1996. ACM Press/ACM SIGGRAPH, New York.

[GME+00] A. Gregory, A. Mascarenhas, S. Ehmann, M. Lin, and D. Manocha. Six degree-of-freedom haptic display of polygonal models. In *IEEE Visualization Conference*, 2000.

[GRLM03] N. Govindaraju, S. Redon, M. Lin, and D. Manocha. Cullide : Interactive collision detection between complex models in large environments using graphics hardware. In *EUROGRAPHICS Workshop on Graphics Hardware*, San Diego, Juillet 2003.

[GSF94] A. Garcia-Alonso, N. Serrano, and J. Flaquer. Solving the detection problem. *IEEE Computer Graphics and Applications*, pages 36–43, Mai 1994.

[Hah88] J.K. Hahn. Realistic animation of rigid bodies. *Computer Graphics (Proceedings of ACM SIGGRAPH 88)*, 22(4) :299–308, Août 1988.

[HBS99] C.H. Ho, C. Basdogan, and M.A. Srinivasan. Efficient point-based rendering techniques for haptic display of virtual objects. *Presence*, 8(5) :477–491, Octobre 1999.

[HBZ90] B. Von Herzen, A.H. Barr, and H.R. Zatz. Geometric collisions for time dependent parametric surfaces. *Computer Graphics (Proceedings of ACM SIGGRAPH 90)*, 24(4) :39–48, Août 1990.

[HDLM96] M. Hughes, C. DiMattia, M.C. Lin, and D. Manocha. Efficient and accurate interference detection for polynomial deformation. In *Computer Animation'96 Conference*, 1996.

[He99] T. He. Fast collision detection using quospo trees. In *Symposium on Interactive 3D Graphics*, pages 55–62, 1999.

[HKM95] M. Held, J. Klosowski, and J.S. Mitchell. Evaluation of collision detection methods for virtual reality fly-throughs. In *Seventh Canadian Conference on computational Geometry*, pages 205–210, Québec City, Août 1995.

[Hub96] P.M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3) :179–209, Juillet 1996.

[HZLM01] K.E. Hoff, A. Zaferakis, M. Lin, and D. Manocha. Fast and simple geometric proximity queries using graphics hardware. In *Symposium on Interactive 3D Graphics*, 2001.

[JC98] D.E. Johnson and E. Cohen. A framework for efficient minimum distance computations. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3678–3684, Louvain, Mai 1998.

[JC99] D.E. Johnson and E. Cohen. Bound coherence for minimum distance computations. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1843–1848, Detroit, Mai 1999.

[JGT89] N.M. Thalmann J.P. Gourret and D. Thalmann. Simulation of object and human skin deformation in a grasping task. *In Computer Graphics (Proc SIGGRAPH)*, 23 :21–30, 1989.

[JSL99] A. Joukhadar, A. Scheuer, and Ch. Laugier. Fast contact detection between moving deformable polyhedra. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Kyongju, Octobre 1999.

[KGS98] D.J. Kim, L.J. Guibas, and S.Y. Shin. Fast collision detection among multiple moving spheres. *IEEE Transactions on Visualisation and Computer Graphics*, 4(3) :228–242, Juillet 1998.

[KHM+98] J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualisation and Computer Graphics*, 4(1), Mars 1998.

[KLM02] Y. Kim, M. Lin, and D. Manocha. Deep : Dual-space expansion for estimating penetration depth between convex polytopes. In *IEEE International Conference on Robotics and Automation (ICRA)*, Washington, Mai 2002.

[KOLM02a] Y. Kim, M. Otaduy, M. Lin, and D. Manocha. Fast penetration depth computation for physically-based animation. In *ACM Symposium on Computer Animation*, San Antonio, Juillet 2002.

[KOLM02b] Y. Kim, M. Otaduy, M. Lin, and D. Manocha. Six degree of freedom haptic display using localized contact computations. In *10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Mars 2002.

[KP03] D. Knott and D. Pai. Cinder : Collision and interference detection in real-time using graphics hardware. In *Graphics Interface'03*, 2003.

[KPLM98] S. Krishnan, A. Pattekar, M.C. Lin, and D. Manocha. Spherical shell : a higher order bounding volume for fast proximity queries. In *3rd International Workshop on Algebraic Foundations and Robotics*, 1998.

[LC87] W. Lorensen and H. Cline. Marching cubes : a high resolution 3d surface construction algorithm. *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, 21(4) :163–169, Juillet 1987.

[LC91] M.C. Lin and J.F. Canny. A fast algorithm for incremental distance calculation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1008–1015, Sacramento, Avril 1991.

[LC92] M.C. Lin and J.F. Canny. Efficient collision detection for animation. In *EUROGRAPHICS Workshop on Computer Animation and Simulation (EGCAS)*, Cambridge, Septembre 1992.

[LCN99] J.C. Lombardo, M.P. Cani, and F. Neyret. Real-time collision detection for virtual surgery. In *Computer Animation'99 Conference*, Genève, Mai 1999.

[LGLM00] E. Larsen, S. Gottshalk, M.C. Lin, and D. Manocha. Fast distance queries with rectangle swept sphere volumes. In *IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, Avril 2000.

[LM95] M.C. Lin and D. Manocha. Fast interference detection between geometric models. *The Visual Computer*, 11 :542–561, 1995.

[LM01] T. Larsson and A. Möller. Collision detection for continously deforming bodies. 20(3), Septembre 2001.

[LMCG96] M.C. Lin, D. Manocha, J. Cohen, and S. Gottshalk. Collision detection : Algorithms and applications. *Algorithmics for Robotics Motion and Manipulation*, pages 129–142, 1996.

[LR80] J. M. Lane and R. Riesenfield. A theoritical developpement for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2 :35–46, 1980.

[Löt81] Per Lötstedt. Coulomb friction in two dimensional rigid body systems. 61 :605–615, 1981.

[LWTC99] B.-S. Lee, C. Wentong, S. Turner, and L. Chen. Adaptive dead reckoning algorithms for distributed interactive simulation. In *13th Workshop on parallel and distributed simulation*, 1999.

[MC94] Brian Mirtich and John Canny. Impulse-based dynamic simulation. In *Proceedings of 1995 Symposium on Interactive 3D Graphics*, 1994.

[MC97] P. Meseure and C. Chaillou. Détection de collisions entre modèles polyédriques : quelques propositions. In *5ème séminaire du groupe de travail Animation et Simulation*, pages 91–100, Reims, Mars 1997.

[MHC98] P. Meseure, L. Hilde, and C. Chaillou. Accélération de la détection de collisions entre corps rigides et déformables. In *Actes des 6èmes journées du Groupe de Travail Réalité Virtuelle*, pages 167–174, Paris, Mars 1998.

[Mil96] Victor J. Milenkovic. Position based physics : Simulating the motion of many highly interacting spheres and polyhedra. In *SIGGRAPH 96 Conference Proceedings*, pages 129–136, 1996.

[Mir96] B. Mirtich. Hybrid simulation : combining constraints and impulses, 1996. Technical Report, Department of Computer Science, University of California, Berkeley.

[Mir97] B. Mirtich. V-clip : Fast and robust polyhedral collision detection. Technical Report TR97-05, Université de Cambridge, juin 1997.

[Mir98] B. Mirtich. Rigid body contact : Collision detection to force computation, 1998. Technical Report TR-98-01, Mitsubishi Electrical Research Laboratory.

[Mir00] Brian Mirtich. Timewarp rigid body simulation. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 193–200. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[Möl97] T. Möller. A fast triangle-triangle intersection test. *Journal of Graphics Tools*, 1997.

[Moo87] P.M. Moore. *A Flexible Object Animation System.* PhD thesis, University of California, Santa Cruz, 1987.

[MPT99] W.A. McNeely, K.D. Puterbaugh, and J.J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. In *SIGGRAPH 99 Computer Graphics Proceedings*, Annual Conference Series, pages 401–408, Los Angeles, Août 1999. ACM Press/ACM SIGGRAPH, New York.

[MS01] V. Milenkovic and H. Schmidl. Optimization-based animation. In *SIGGRAPH 2001 Conference Proceedings*, Computer Graphics annual conference series, pages 37–46, Los Angeles, Août 2001. ACM Press/ACM SIGGRAPH, New York.

[Mur97] Katta G. Murty. *Linear Complementarity, Linear And Nonlinear Programming.* Internet Edition, 1997.

[MW88] M. Moore and J. Wilhelms. Collision detection and response for computer animation. *Computer Graphics (Proceedings of ACM SIGGRAPH 88)*, 22(4) :289–298, Août 1988.

[MZ90] M. McKenna and D. Zeltzer. Dynamic simulation of autonomous legged locomotion. *In Computer Graphics (Proc. SIGGRAPH)*, 24 :29–38, 1990.

[NAT90] B. Naylor, J. Amanatides, and W. Thibault. Merging bsp trees yields polyhedral set operations. *Computer Graphics (Proceedings of ACM SIGGRAPH 90)*, 24(4) :115–124, Août 1990.

[OD99] C. O'Sullivan and J. Dingliana. Real-time collision detection and response using sphere-tree. In *15th Spring Conference on Computer Graphics*, pages 83–92, Budmeria, Avril 1999.

[OvdS96] M. Overmars and A.F. van der Stappen. Range searching and point location among fat objects. *Journal of Algorithms*, 21(3), Novembre 1996.

[PG95] I.J. Palmer and R.L. Grimsdale. Collision detection for animation using sphere-trees. *Computer Graphics forum*, 14(2) :105–116, 1995.

[PML95] M.K. Ponamgi, D. Manocha, and M.C. Lin. Incremental algorithms for collision detection between solid models. In *ACM/SIGGRAPH Symposium on Solid Modeling*, pages 293–304, 1995.

[Pro97] X. Provot. Collision and self-collision handling in cloth model dedicated to design garments. In *Graphics Interface'97 Conference*, pages 177–189, Kelowna, Mai 1997.

[Qui94] S. Quinlan. Efficient distance computation between non-convex objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1994.

[RKC00] S. Redon, A. Kheddar, and S. Coquillart. An algebraic solution to the problem of collision detection for rigid polyhedral objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, Avril 2000.

[RKC01] S. Redon, A. Kheddar, and S. Coquillart. Contact : In-between motions for collision detection. In *IEEE International Workshop on Robot and Human Interactive Communication (ROMAN)*, 2001.

[RKC02a] S. Redon, A. Kheddar, and S. Coquillart. Fast continuous collision detection between rigid bodies. *Computer Graphics Forum (Proceedings of EUROGRAPHICS'02)*, 21(3), Septembre 2002.

[RKC02b] S. Redon, A. Kheddar, and S. Coquillart. Hierarchical back-face culling for collision detection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Octobre 2002.

[RKCS92] Jong-Shi Pang Richard K. Cottle and Richard E. Stone. *The Linear Complementarity Problem*. Academic Press, 1992.

[RKK97] D.C. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. In *SIGGRAPH 97 Computer Graphics Proceedings*, Computer Graphics Annual conference series, pages 345–352, Los Angeles, Août 1997. ACM Press/ACM SIGGRAPH, New York.

[Sch02] Harald Schmidl. *Optimization Based Animation*. PhD thesis, University of Miami, 2002.

[SdM00] K. Sundaraj, D. d'Aulignac, and E. Mazer. A new algorithm for computing minimum distance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2000.

[Sei90] R. Seidel. Linear programming and convex hulls made easy. In *Sixth Annual Symposium on Computational Geometry*, pages 211–215, 1990.

[SKTK95] A. Smith, Y. Kitamura, H. Takemura, and F. Kishino. A simple and efficient method for accurate collision detection among deformable polyhedral objects in arbitrary motion. In *IEEE Virtual Reality Annual International Symposium*, pages 136–145, Mars 1995.

[SL00] K. Sundaraj and C. Laugier. Fast contact localisation of moving deformable polyhedras. In *IEEE International Conference on Automation, Robotics, Control and Vision*, 2000.

[SN90] T. Sederberg and T. Nishita. Curve intersection using Bézier clipping. *Computer Aided Design*, 22(9) :538–549, 1990.

[SP91] S. Sclaroff and A. Pentland. Generalized implicit functions for computer graphics. *Computer Graphics (Proceedings of ACM SIGGRAPH 91)*, 25(4) :247–250, Juillet 1991.

[SP95] V.V. Savchenko and A.A. Pasko. Collision detection for functionally defined deformable objects. In *EUROGRAPHICS Workshop on Implicit Surfaces*, pages 217–221, Grenoble, Avril 1995.

[SS98] J. Sauer and E. Schömer. A constraint based approach to rigid body dynamics for virtual reality applications, 1998. Proceedings of the ACM Symposium on Virtual Reality Software and Technology.

[ST96] D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid-body dynamics with inelastic collisions and coulomb friction. In *International J. Numerical Methods in Engineering*, volume 39, 1996.

[ST97] D.E. Stewart and J.C. Trinkle. Dynamics, friction, and complementarity problems. pages 425–439, 1997.

[Ste00] D.E. Stewart. Rigid-body dynamics with friction and impact. *SIAM Review*, 42(1) :3–39, 2000.

[SWF+93] J. M. Snyder, A.R. Woodbury, K. Fleisher, B. Currin, and A.H. Barr. Interval methods for multi-point collisions between time-dependent curved surfaces. In *SIGGRAPH 93 Conference Proceedings*, Computer Graphics annual conference series, pages 312–334, Anaheim, Août 1993. ACM Press/ACM SIGGRAPH, New York.

[TC96] C. Tzafestas and P. Coiffet. Real-time collsion detection using spherical octrees : Virtual reality application. In *IEEE International Workshop on Robot and Human Communication*, Tsukuba, 1996.

[TC98] C. Turnbull and S. Cameron. Computing distances between nurbs-defined convex objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3686–3690, 1998.

[TF88] D. Terzopoulos and K. Fleisher. Deformable models. *Visual Computer*, 4 :306–331, 1988.

[Tha00] U. Thatcher. *Loose Octrees, Game Programming Gems*, chapter 4.11. Mark Delowa Edt., Charles River Media, Rockland, Massachusetts, 2000.

[TN87] W. Thibault and B. Naylor. Set operations on polyhedra using binary space partitioning trees. *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, 21(4) :153–162, Juillet 1987.

[TPB87] D. Terzopoulos, J.C. Platt, and A.H. Barr. Elastically deformable models. *In Computer Graphics (Proc. SIGGRAPH)*, 21 :205–214, 1987.

[TTRC00] F. Thomas, C. Turnbull, L. Ros, and S. Cameron. Computing signed distances between free-form objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3713–3718, San Fransisco, Avril 2000.

[Tur89] G. Turk. Interactive collision detection for molecular graphics. Technical Report TR90-014, University of North Carolina, Master Thesis, 1989.

[Van91] G. Vanecek. Brep-index : a multidimensional space partitioning tree. *International Journal of Computational Geometry and Applications*, 1(3) :243–261, 1991.

[Van94] G. Vanecek. Back face culling applied to collision detection of polyhedra. *The Journal of Visualization and Computer Animation*, 5, 1994.

[VT94] P. Volino and N.M. Thalmann. Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. *Computer Graphics Forum (Proceedings of EUROGRAPHICS'94)*, 13(3) :155–166, Septembre 1994.

[VT95] P. Volino and N.M. Thalmann. Collision and self-collision detection : Efficient and robust solutions for highly deformable surfaces. In *EUROGRAPHICS workshop on Computer Animation and Simulation (EGCAS)*, pages 55–65, Maastricht, Septembre 1995.

[Wil87] J. Wilhelms. Toward automatic motion control. *IEEE Computer Graphics And Application*, 7 :11–22, 1987.

[WLML99] A. Wilson, E. Larsen, D. Manocha, and M.C. Lin. Partitioning and handling massive models for interactive collision detection. *Computer Graphics Forum (Proceedings of EUROGRAPHICS'99)*, 18(3) :319–330, Septembre 1999.

[WNDS99] M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL Programming Guide : the Official Guide to Learning OpenGL version 1.2.* Addison Wesley, 3ème edition, 1999.

[Zac94] G. Zachmann. Exact and fast collision detection. Master's thesis, Darmstadt University of Technology, Department of Computer Science, 1994.

[Zac97] G. Zachmann. Real-time and exact collision detection for interactive virtual prototyping. In *ASME Design Engineering Technical Conferences'97 Conference*, Sacramento, Septembre 1997.

[Zac98] G. Zachmann. Rapid collision detection by dynamically aligned dop-trees. In *IEEE Virtual Reality Annual International Symposium'98*, Atlanta, Mars 1998.

[Zac01] G. Zachmann. Optimizing the collision detection pipeline. In *1st International Game Technology Conference*, Janvier 2001.

[ZK03] G. Zachmann and G. Knittel. An architecture for hierarchical collision detection. *Journal of WSCG (Proceedings of WSCG'03)*, 11(1) :149–156, Février 2003.

[ZPG95] M. Zeiller, W. Purgathofer, and M. Gervautz. Efficient collision detection for general csg objects. In *EUROGRAPHICS workshop on Computer Animation and Simulation (EGCAS)*, pages 66–79, Maastricht, Septembre 1995.