Algorithmes de simulation dynamique interactive d'objets rigides

Stéphane Redon

N° d'ordre: 02EVRY0009

UNIVERSITÉ D'EVRY

THÈSE

présentée pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ D'EVRY
Spécialité INFORMATIQUE ET ROBOTIQUE

par

Stéphane REDON

ALGORITHMES DE SIMULATION DYNAMIQUE INTERACTIVE D'OBJETS RIGIDES

soutenue le jeudi 10 octobre 2002 devant un jury composé de

M. Philippe Fuchs	Président
Mme Sabine Coquillart	Directeurs de thèse
M. Abderrahmane Kheddar	
Mme Marie-Paule Cani	Rapporteurs
M. Philippe Coiffet	
Mme Ming C. Lin	
M. Claude Andriot	Examinateurs
M. Michel Dureigne	

Remerciements

Les travaux présentés dans ce mémoire ont été réalisés au sein de l'Action i3D de l'INRIA, en collaboration avec le CEMIF-SC de l'université d'Evry.

Je tiens tout d'abord à remercier chaleureusement mes deux directeurs de thèse, Madame Sabine Coquillart, directrice de recherche à l'INRIA, pour m'avoir intégré au coeur de son équipe dynamique, et avoir su créer un environnement extrèmement favorable à l'avancement des travaux de la thèse, et Monsieur Aberrahmane Kheddar, maître de conférences à l'université d'Evry, pour son expertise en automatique et en haptique, sa très grande pédagogie et ses qualités humaines. Je les remercie tous deux de la confiance qu'ils ont bien voulu m'accorder.

J'aimerais remercier ensuite les trois rapporteurs de cette thèse, Madame Marie-Paule Cani, professeur à l'Institut National Polytechnique de Grenoble, Monsieur Philippe Coiffet, directeur de recherche au CNRS, et Madame Ming C. Lin, professeur associé à l'Université de Caroline du Nord à Chapel Hill. Leurs remarques et encouragements m'ont particulièrement aidé à franchir le dernier cap de la thèse.

Je remercie également les examinateurs du jury de thèse. Monsieur Philippe Fuchs, Maître de conférences à l'ENSMP, pour avoir bien voulu présider le jury de thèse. Monsieur Claude Andriot, du CEA LIST, avec qui j'ai eu la chance de travailler lors de l'application des travaux à la simulation avec retour d'efforts. Monsieur Michel Dureigne, du Centre Commun de Recherche d'EADS, pour sa vision industrielle du domaine, et des discussions extrèmement intéressantes sur le concept de physique virtuelle.

J'aimerais aussi remercier mes compagnons d'i3D. Nicolas Tarrin et Tangui Morvan, pour leur aide essentielle et efficace lors du portage de CONTACT sur le plan de travail virtuel, pour la simulation avec retour d'efforts, et pour le report (un peu trop) constant de bugs en tous genres. Monsieur Robert Ehrlich, également, pour son immense savoir technique et son aide précieuse lors de problèmes systèmes récalcitrants. Aussi, un grand merci à Alexis, Anatole, Boris, Jérôme, Mathieu et Terii. J'ai beaucoup apprécié le travail (et les pauses) en votre compagnie.

Un grand merci, aussi, à mes amis Arnaud (qui doit être remercié également pour sa relecture de ce mémoire), David B., David C., Frédéric, Laurent, Nicolas, Pierre-Yves, et Alexandre, pour leur soutien constant, et parce qu'ils sont, tout simplement, mes amis.

Enfin, je remercie et j'embrasse ma famille, pour leur amour, leur soutien moral, leurs encouragements et leur présence constante.

Avant-Propos

Notre travail porte sur le développement de simulateurs dynamiques interactifs d'objets rigides.

Dans de tels simulateurs, les deux principaux problèmes à résoudre, à savoir la *détection de collisions* entre les objets virtuels et le *calcul de leur mouvement*, prennent une coloration nouvelle en raison de l'intégration de l'homme dans la boucle de simulation : d'une part, les algorithmes permettant de résoudre ces deux principaux problèmes doivent être suffisamment *efficaces* pour permettre une réponse rapide lorsque l'utilisateur intervient dans la simulation et, d'autre part, les phénomènes simulés (ici les mouvements d'objets rigides) doivent être *réalistes*.

L'objet de ce mémoire est de proposer des algorithmes pour résoudre chacun des deux problèmes posés par le développement d'un simulateur dynamique interactif d'objets rigides.

La majeure partie des méthodes de détection de collisions sont *discrètes* : elles détectent seulement des interpénétrations entre les objets à des instants successifs. Ceci peut présenter un certain nombre d'inconvénients, dont le manque de réalisme, que les méthodes *continues*, plus rares, qui calculent l'instant de premier contact entre les objets, parviennent à résoudre au prix toutefois d'un temps de calcul plus important.

Nous proposons des algorithmes de détection de collisions continue efficaces, permettant d'interagir avec des objets virtuels complexes. Nous proposons également une méthode permettant d'exploiter le mouvement des objets pour accélérer la détection de collisions.

La plupart des méthodes classiques de calcul de mouvement contraint sans frottement sont formulées dans l'espace des contacts. Nous les comparons à une méthode formulée dans l'espace des mouvements, dérivée du principe des moindres contraintes de Gauss, et montrons que, bien que les deux formulations soient mathématiquement équivalentes, la formulation établie dans l'espace des mouvements est plus avantageuse sur le plan algorithmique. Ceci nous incite à proposer un modèle de frottements dans l'espace des mouvements.

Enfin, nous présentons à la fin de ce mémoire plusieurs applications des algorithmes proposés, notamment à des cas industriels fournis par Renault et Airbus-EADS, ainsi que leur utilisation dans la simulation avec retour d'efforts.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Problématique de la simulation dynamique interactive

Dans ce chapitre d'introduction, nous présentons les principes généraux d'un simulateur dynamique interactif d'objets rigides. Nous recensons les principales méthodes apportant des solutions aux deux problèmes posés par un simulateur dynamique : la détection de collisions entre objets virtuels, et le calcul du mouvement des objets. Nous résumons également, à la fin de ce chapitre, les principales contributions de cette thèse.

1.1 Introduction

La simulation dynamique interactive s'inscrit dans une longue lignée de développement majeurs en informatique. Aux progrès spectaculaires de l'infographie, capables de représenter toujours plus de phénomènes physiques et d'afficher des images toujours plus réalistes, ont succédé des avancées fondamentales en simulation dynamique *hors ligne*, dans un premier temps, puis *interactive*.

Au sens large, la simulation dynamique consiste à synthétiser les phénomènes physiques *dépendant du temps*. Les phénomènes simulés peuvent concerner, par exemple, les mouvements des objets rigides ou déformables, la formation et le mouvement des nuages, de l'eau, de la neige ou, à une autre échelle, l'interaction entre des atomes dans une molécule.

En infographie, il s'agit essentiellement de simuler l'aspect *visuel* des phénomènes physiques dans des animations. Dans un environnement virtuel, cependant, pouvant inclure d'autres canaux sensoriels que la vision, il peut être également souhaitable de simuler les aspects *sonores*, *olfactifs* et *tactiles* des phénomènes physiques que l'on cherche à représenter. Lorsqu'un utilisateur peut intervenir sur les phénomènes physiques simulés dans l'environnement virtuel, la simulation dynamique devient *interactive*, ce qui impose, bien sûr, que les algorithmes de simulation soient suffisamment

efficaces pour permettre une réaction rapide de l'environnement virtuel dès que l'utilisateur y intervient.

Dans ce mémoire, nous allons nous concentrer uniquement sur la *simulation interactive d'objets rigides*. Comme nous le verrons un peu plus loin, ce domaine à déjà fait l'objet de nombreuses publications. L'intérêt porté aux simulations interactives d'objets rigides s'explique par leurs nombreuses applications, par exemple en infographie (pour obtenir des animations réalistes), en robotique (pour la planification de mouvements, la télé-opération, la simulation de chaînes d'assemblage, la simulation du retour d'efforts), dans l'industrie (pour le prototypage virtuel, les simulations de montage/démontage), et plus récemment les jeux vidéo. Certains simulateurs dynamiques sont ainsi distribués, soit en combinaison avec d'autres produits (par exemple lorsqu'ils sont intégrés dans un logiciel de modélisation et d'animation), soit sous la forme de librairies indépendantes (parfois gratuites), intégrables dans d'autres applications.

Les simulations d'objets rigides interactives constituent toutefois un sujet de recherche encore actif, essentiellement à cause du compromis nécessaire entre le réalisme de la simulation et l'efficacité des algorithmes permettant une interactivité satisfaisante.

La suite du chapitre est organisée de la manière suivante. La section 1.2 présente brièvement le schéma général d'un simulateur dynamique interactif d'objets rigides. Nous verrons dans cette section qu'un simulateur dynamique comporte deux modules principaux : le premier pour détecter les collisions entre objets virtuels, et le second pour calculer le mouvement des objets. Les sections 1.3 et 1.4 présentent respectivement les principales méthodes connues de détection de collisions et celles de calcul du mouvement des objets. Enfin, la section 1.5 résume les principales contributions de cette thèse.

1.2 Schéma général d'un simulateur dynamique

Afin d'expliquer brièvement le fonctionnement d'un simulateur dynamique interactif d'objets rigides par contraintes, décrit dans la figure 1.1, nous allons considérer le cas d'un seul objet rigide mobile manipulé par un utilisateur dans un environnement virtuel statique. Cet objet est soumis à des *forces extérieures*, telles la gravité, le vent, la résistance de l'air, les actions de l'utilisateur, et à des *forces de réaction*, créées par les contraintes géométriques exercées sur l'objet. Ces contraintes peuvent être *unilatérales*, lorsqu'elles représentent des contacts provisoires entre objets, ou *bilatérales*, et donc permanentes, lorsqu'elles traduisent des articulations entre objets. Aussi les forces extérieures sont-elles, à chaque instant, les *seules* informations fournies par l'utilisateur d'un simulateur dynamique.

Les équations d'Euler-Newton décrivant le mouvement d'un solide sont données

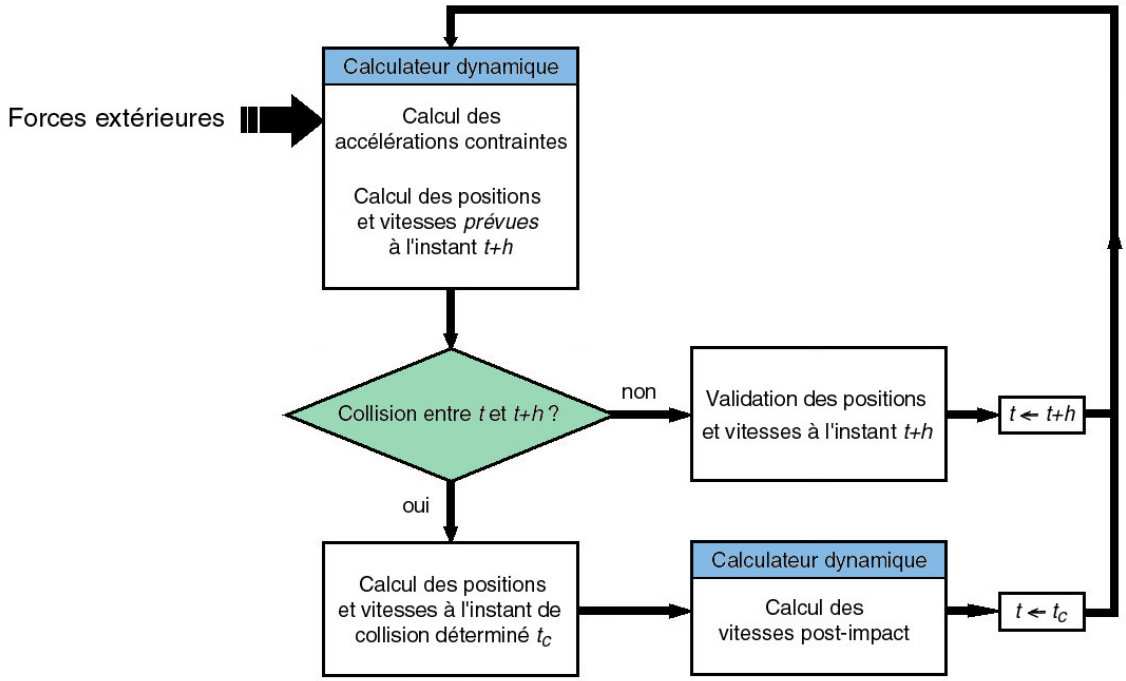


Fig. 1.1 – Schéma d'un simulateur dynamique d'objets rigides par contraintes.

par [WB97, BHB99] :

$$\frac{d}{dt}\begin{pmatrix}\mathbf{x}(t)\\ \mathbf{R}(t)\\ \mathbf{P}(t)\\ \mathbf{L}(t)\end{pmatrix} = \begin{pmatrix}\mathbf{v}(t)\\ \omega(t)\mathbf{R}(t)\\ \mathbf{F}(t)\\ \tau(t)\end{pmatrix} \tag{1.1}$$

avec

$$\begin{cases}\mathbf{v}(t) = \frac{\mathbf{P}(t)}{M}\\ \omega(t) = \mathbf{I}(t)\mathbf{L}(t)\\ \mathbf{I}(t) = \mathbf{R}(t)\mathbf{I}_o\mathbf{R}(t)^T\end{cases} \tag{1.2}$$

où $\mathbf{x}$ est la position du centre de gravité du solide, $\mathbf{R}$ est la matrice 3×3 décrivant son orientation, $\mathbf{P}$ sa quantité de mouvement, $\mathbf{L}$ son moment angulaire, $\mathbf{v}$ sa vitesse translationnelle, ω sa vitesse rotationnelle, $\mathbf{F}$ la somme des forces qui lui sont appliquées et τ le couple total qui lui est appliqué. La masse M est constante et le tenseur d'inertie dans le repère global $\mathbf{I}$ est obtenu à partir du tenseur d'inertie du solide dans le repère local du solide $\mathbf{I}_o$.

Ces équations différentielles sont continues. Dans un simulateur, cependant, l'on ne souhaite connaître l'état de l'environnement virtuel qu'à certains instants discrets, par exemple les instants auxquels une nouvelle image est affichée. Plus précisément, nous souhaitons connaître les *positions*, *vitesses* et *accélérations* des objets pour une succession d'instants discrets t_n, pour $n \geqslant 0$. Notons d'ailleurs que, dans le cas d'un mouvement contraint général il est impossible, la plupart du temps, de donner une expression analytique de la solution des équations (1.1) et (1.2).

Supposons par exemple que l'instant courant dans la simulation est $t_n = t$. A cet instant, la position et la vitesse de l'objet sont connues, ainsi que les forces extérieures qui lui sont appliquées. De plus, le simulateur connaît toutes les contraintes, unilatérales ou bilatérales, exercées sur l'objet. En fonction des forces extérieures et des contraintes, le *calculateur dynamique* du simulateur peut déterminer les forces de réaction engendrées par les liaisons, et en déduire *l'accélération contrainte* de l'objet à l'instant t. Il s'agit du premier des deux problèmes dynamiques que doit être capable de résoudre un simulateur : le *problème du calcul du mouvement contraint*. Dans la figure 1.2.a, les forces extérieures appliquées sur le cube en son centre de gravité sont le poids $\mathbf{p}$ et la force exercée par l'utilisateur $\mathbf{f}_u$. En supposant que la masse du cube est égale à 1, l'accélération non contrainte du cube, c'est-à-dire l'accélération que le cube aurait si le sol n'était pas là, est $\mathbf{a}_u$. Le calculateur dynamique détermine qu'à cause du sol, l'accélération contrainte du cube est $\mathbf{a}$.

A partir de cette accélération contrainte, le simulateur peut maintenant déterminer la position et la vitesse de l'objet au prochain instant $t_{n+1} = t_n + h$, où h est la durée du pas de temps de la simulation, en discrétisant les équations de la dynamique (1.1) et (1.2) et en utilisant un schéma d'intégration particulier [WB97]. Cette position et cette vitesse sont toutefois seulement *prévues*, puisqu'elles correspondent à la position et à la vitesse de l'objet à l'instant t_{n+1} *si aucune collision n'intervient entre* t_n *et* t_{n+1}.

Le module de détection de collisions, deuxième module principal d'un simulateur dynamique avec le calculateur dynamique, détermine donc en fonction du *mouvement intentionnel* de l'objet si une collision intervient entre t_n et t_{n+1}.

Si ce n'est pas le cas, alors les position et vitesse intentionnelles de l'objet sont validées et deviennent les position et vitesse *réelles* de l'objet à l'instant $t_{n+1} = t + h$. Le nouvel instant courant de la simulation est alors $t + h$, et le simulateur met à jour les contraintes géométriques exercées sur l'objet (les points de contact ou les liaisons avec les autres objets) et peut boucler, c'est-à-dire effectuer les calculs pour ce nouvel instant.

Si, par contre, une collision est détectée à un instant t_c compris dans l'intervalle de temps $[t_n, t_{n+1}]$, alors le simulateur détermine les position et vitesse de l'objet à cet instant t_c, et met à jour l'ensemble des contraintes géométriques. En particulier, une contrainte unilatérale est ajoutée à l'ensemble des contraintes agissant sur l'objet pour représenter le nouveau contact.

A l'instant de collision t_c, cependant, la vitesse locale de l'objet au point de contact est incompatible avec la contrainte nouvellement créée. Si cette vitesse n'est pas modifiée, en effet, l'objet pénètrera l'environnement virtuel à l'instant suivant. Autrement dit, l'objet doit *rebondir* sur l'environnement virtuel (au sens large, la vitesse de rebond pouvant être nulle). Le calculateur dynamique doit donc, en fonction des propriétés physiques des objets et de la vitesse d'impact (*i.e.* en t_c^-), calculer la vitesse de l'objet immédiatement après l'impact (*i.e.* en t_c^+). Ceci est le second problème dynamique qu'un simulateur doit être capable de résoudre : le *problème du calcul de*

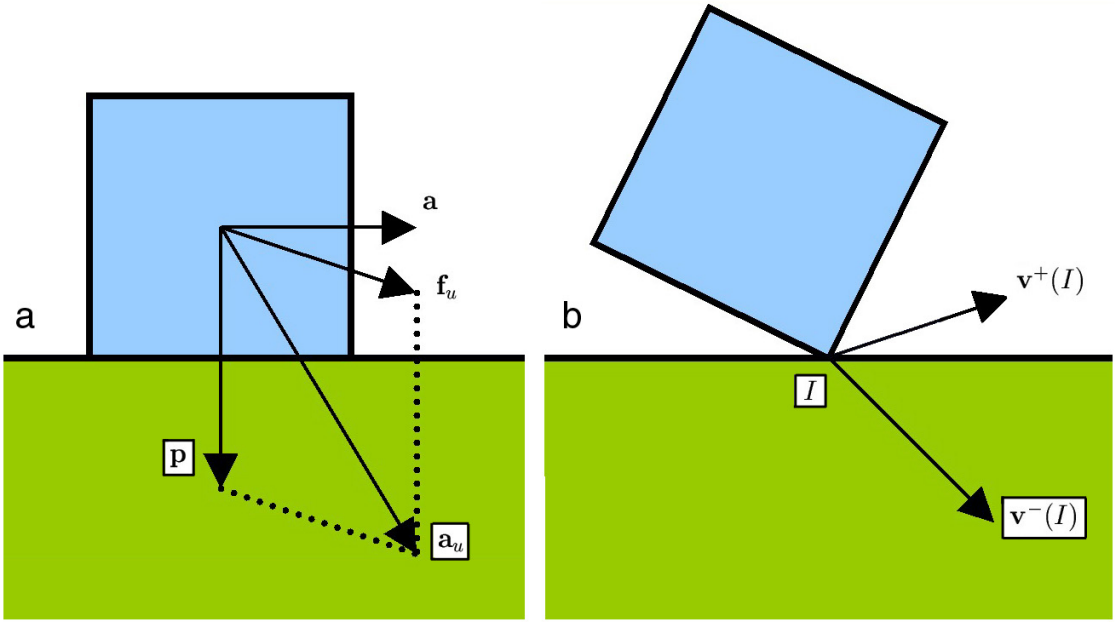


Fig. 1.2 – Les deux problèmes dynamiques que doit pouvoir résoudre un simulateur. a : Le calcul du mouvement contraint. b : Le calcul de la réponse à la collision. Dans le problème du calcul du mouvement contraint, le simulateur doit déterminer l'accélération contrainte $\mathbf{a}$ de l'objet en fonction de son accélération non contrainte $\mathbf{a}_u$, dépendant de son poids $\mathbf{p}$ et de la force exercée par l'utilisateur $\mathbf{f}_u$. Dans le problème du calcul de la réponse à la collision, le simulateur doit déterminer la vitesse post-impact $\mathbf{v}^+$ de l'objet en fonction de sa vitesse pre-impact $\mathbf{v}^-$ et des propriétés des objets en contact.

la réponse à la collision. Ainsi, alors que les accélérations des objets sont continues à chaque instant, les vitesses des objets peuvent être discontinues lors de collisions. Dans la figure 1.2.b, la vitesse $\mathbf{v}^-(I)$ du point du cube I qui vient d'entrer en collision avec l'obstacle à l'instant t_c^- doit être modifiée pour que le cube ne pénètre pas dans le sol à l'instant suivant. Une vitesse post-impact valide serait par exemple $\mathbf{v}^+(I)$.

Une fois la vitesse post-impact de l'objet calculée, le nouvel instant courant de la simulation devient $t_{n+1} = t_c$, et non l'instant initialement prévu $t_n + h$. Le simulateur met à jour les contraintes géométriques pour ce nouvel instant et peut boucler.

Ainsi, un simulateur dynamique contient essentiellement deux modules. Le premier détecte les collisions entre les objets virtuels, alors que le second calcule les réactions aux collisions, en déterminant soit les accélérations contraintes des objets, soit leurs vitesses post-impact. Dans la section suivante, nous allons nous intéresser plus en détail aux principales méthodes existantes de détection de collisions.

1.3 Méthodes de détection de collisions

Il est possible de répartir les méthodes de détection de collisions en trois grandes catégories :

- les méthodes *discrètes* : il s'agit de la grande majorité des méthodes. Elles échantillonnent la trajectoire des objets et ne détectent que des *interpénétrations* entre objets à certains instants.
- les méthodes *pseudo-continues* : elles détectent des recouvrements entre les volumes balayés par les objets au cours d'intervalles de temps. Cependant, la projection des trajectoires dans l'espace 3d entraîne la perte de l'information temporelle, et ces méthodes ne peuvent pas directement déterminer l'instant de premier contact. De plus, la projection des trajectoires est souvent difficile à déterminer efficacement.
- les méthodes *continues* : elles calculent l'instant de premier contact entre objets *pendant* la détection de collisions. Ce calcul est partie intégrante de l'algorithme.

Ce mémoire concerne essentiellement les objets polyédriques rigides (hormis l'extension aux surfaces paramétrées ou implicites rigides proposée au chapitre 3). Pour de tels objets, la détection de collisions discrète revient à détecter les paires de triangles qui s'interpénètrent.

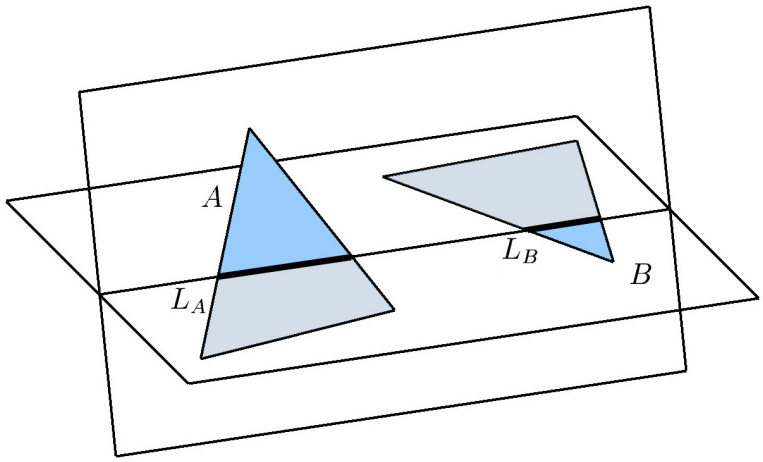


Fig. 1.3 – Les deux triangles s'interpénètrent si et seulement si les intervalles L_A et L_B se recouvrent.

Plusieurs méthodes existent pour détecter de telles paires [Hel97, Mol97]. Celle présentée par Möller, par exemple, et visible dans la figure 1.3, consiste à examiner les positions relatives des intervalles L_A et L_B déterminés par l'intersection des triangles A et B avec la droite caractérisant l'intersection des plans contenant les triangles (le cas particulier des triangles parallèles est résolu séparément).

La détection de collisions continue entre objets polyédriques est conceptuellement assez simple, bien que plus coûteuse que la détection de collisions discrète. En effet,

les méthodes continues ne doivent détecter que trois types de contact non dégénérés : tous les contacts (et non pas les interpénétrations) entre deux objets polyédriques A et B font ainsi intervenir au moins l'un des trois types de contact élémentaires non dégénérés :

- une arête de A entre en contact avec une arête de B.
- un point de A entre en contact avec une face de B.
- une face de A entre en contact avec un point de B.

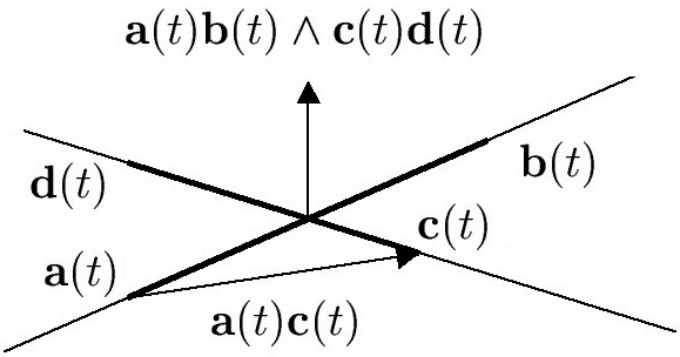


Fig. 1.4 – détection de collisions entre deux arêtes.

Ces types de contact sont simples à exprimer géométriquement. Ainsi, pour le cas arête/arête, il suffit de détecter une collision entre les droites contenant les arêtes. Si $\mathbf{a}(t)\mathbf{b}(t)$ est la première arête et $\mathbf{c}(t)\mathbf{d}(t)$ est la seconde arête, alors les droites entrent en contact lorsque :

$$\mathbf{a}(t)\mathbf{c}(t) \cdot (\mathbf{a}(t)\mathbf{b}(t) \wedge \mathbf{c}(t)\mathbf{d}(t)) = 0 \tag{1.3}$$

c'est-à-dire lorsque le vecteur $\mathbf{a}(t)\mathbf{c}(t)$ est dans le plan vectoriel défini par les deux arêtes (*cf* figure 1.4). Une fois qu'un point d'intersection à été détecté à un instant donné, il suffit de déterminer s'il appartient aux arêtes.

Pour les cas point/face et face/point, une collision est d'abord détectée entre le point et le plan support de la face. Si $\mathbf{a}(t)$ est le point $\mathbf{b}(t)\mathbf{c}(t)\mathbf{d}(t)$ le triangle, une collision intervient lorsque :

$$\mathbf{a}(t)\mathbf{b}(t) \cdot (\mathbf{b}(t)\mathbf{c}(t) \wedge \mathbf{b}(t)\mathbf{d}(t)) = 0 \tag{1.4}$$

c'est-à-dire lorsque le vecteur $\mathbf{a}(t)\mathbf{b}(t)$ est dans le plan vectoriel défini par la normale à la face $\mathbf{b}(t)\mathbf{c}(t) \wedge \mathbf{b}(t)\mathbf{d}(t)$ (*cf* figure 1.5). Lorsqu'une telle collision est détectée, il suffit ensuite de déterminer si le point appartient à la face.

Outre le plus grand coût des méthodes de détection de collisions continues, il est facile d'expliquer, au vu du fonctionnement général d'un simulateur dynamique, pourquoi les méthodes discrètes sont utilisées dans la très grande majorité des cas. Dans la plupart des cas, en effet, *les positions des objets ne sont connues qu'à des instants discrets*, les instants t_n, et ce pour deux raisons :

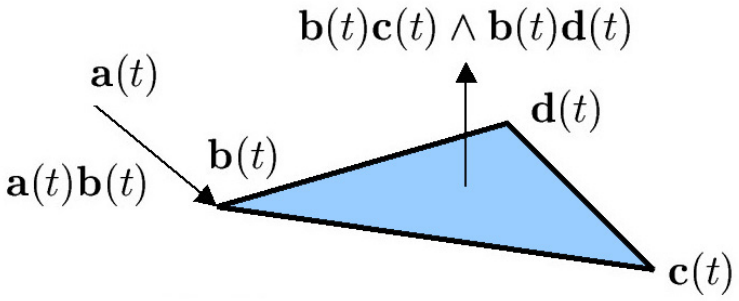


Fig. 1.5 – détection de collisions entre un point et une face.

– un des objets est manipulé par un utilisateur. Comme le périphérique employé n'envoie les actions de l'utilisateur qu'à des instants discrets, il est impossible de savoir ce que fait l'utilisateur entre ces instants. Il est par conséquent impossible de connaître le mouvement de l'objet manipulé.
– les équations de la dynamique sont discrétisées pour être résolues. Dans la plupart des cas où les objets peuvent être contraints par d'autres objets, il n'est pas possible d'obtenir une formulation analytique de leur mouvement, qui n'est représentable qu'*implicitement*, comme solution des équations différentielles (1.1) et (1.2). Le simulateur, par conséquent, doit discrétiser les équations de la dynamique pour estimer les positions successives des objets. Il est alors impossible de connaître exactement le mouvement des objets entre les instants discrets utilisés par le simulateur.

Dans d'autres cas, le mouvement de l'objet est disponible mais est trop complexe, et conduit à des équations de détection de collisions difficiles à résoudre efficacement. Si l'on suppose par exemple que le mouvement de l'objet est connu mais qu'il est représenté par un polynôme de degré très élevé, alors l'évaluation de ce polynôme lors de la résolution des équations de détection de collisions continues (1.3) et (1.4), par exemple par une méthode de dichotomie, peut être très coûteuse et ralentir par trop la résolution.

Dans le chapitre 2, ces points seront utilisés pour justifier l'introduction et l'utilisation de *mouvements intermédiaires arbitraires*, destinés à interpoler les positions successives des objets tout en permettant une résolution efficace des équations de détection de collisions continue.

1.3.1 détection de collisions discrète

Il existe de très nombreuses méthodes de détection de collisions discrètes, tant il s'agit d'un problème fondamental dans de nombreux domaines. Aussi, nous ne donnons ici que les plus connues d'entre elles, qui vont nous permettre de dégager les

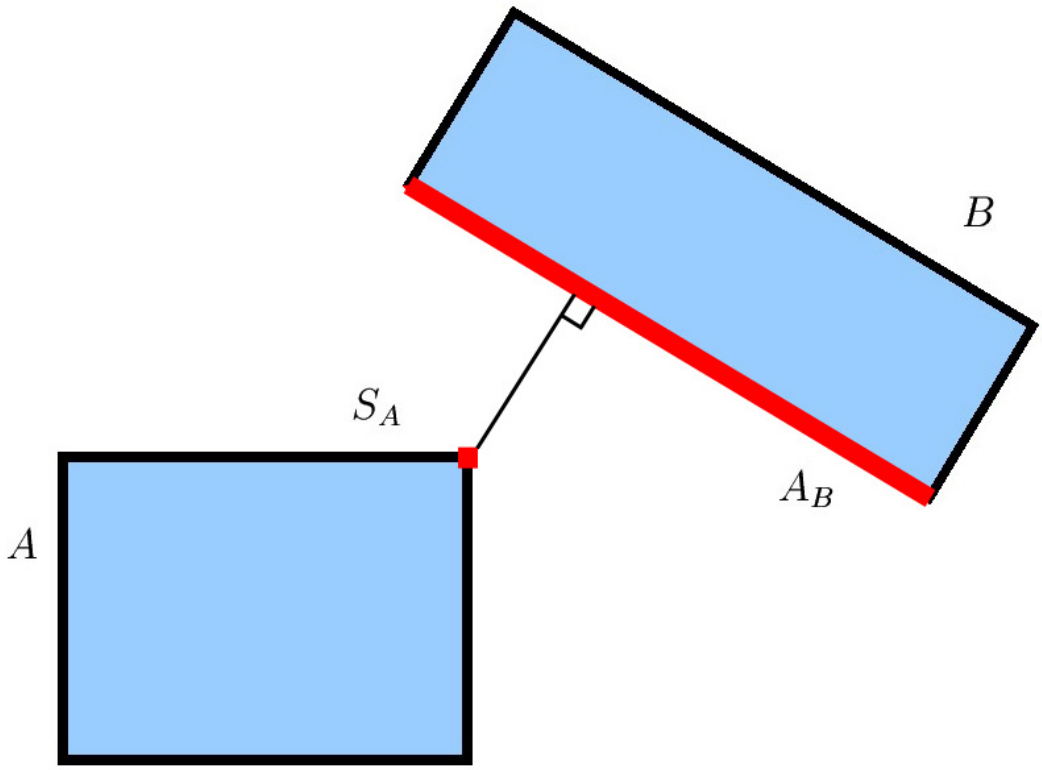


Fig. 1.6 – Le sommet S_A est l'élément de A le plus proche de l'objet B, et l'arête A_B est l'élément de B le plus proche de l'objet A.

principes généraux, et renvoyons le lecteur à d'autres états de l'art plus détaillés [Lin97, LG98, JTT01]. Par ailleurs, ce mémoire ne traitant pour l'essentiel que des objets polyédriques rigides, nous allons *principalement* nous intéresser à ce type d'objet, et renvoyer le lecteur aux mêmes états de l'art pour d'autres représentations d'objets. Souvent, cependant, les principes sont transposables (par exemple, les hiérarchies de volumes englobants sont utilisables quelle que soit la représentation de l'objet).

Notons tout d'abord que plusieurs approches ont été proposées par la communauté de géométrie algorithmique. Par exemple, Dobkin et Kirkpatrick ont proposé un algorithme permettent de déterminer en temps linéaire la séparation de deux polyèdres convexes [DK85a]. D'une autre façon, il est possible de démontrer que la séparation de deux polyèdres convexes peut être déterminée en temps linéaire en réduisant le problème à un problème de programmation linéaire [SK92], qui peut alors être résolu en temps linéaire [Meg83, Dye84].

Le but recherché en géométrie algorithmique, cependant, est généralement d'aboutir à un algorithme dont la complexité *théorique* est la plus faible possible. Ces algorithmes sont parfois difficilement implantables, ou bien présentent des constantes multiplicatives très élevées. L'optique est donc différente en infographie ou en réalité virtuelle, où les algorithmes doivent être utilisables *en pratique*. Il est alors fréquent

d'accorder beaucoup d'attention à l'implantation des algorithmes, et en particulier au nombre d'opérations élémentaires (additions, soustractions, multiplications et divisions) impliquées [GLM96, KHMSZ98].

De nombreux algorithmes, cependant, ont leurs racines dans la géométrie algorithmique. Une approche traditionnelle, par exemple, consiste à restreindre la généralité des modèles géométriques. Il apparaît ainsi qu'il est plus simple de détecter des collisions entre objets polyédriques convexes qu'entre objets polyédriques quelconques.

Appelons *éléments* les primitives polyédriques composant les objets, c'est-à-dire leurs sommets, arêtes et faces. Lorsque deux objets convexes A et B sont disjoints, il est possible de déterminer de façon unique les deux plus proches éléments qui les séparent. Plus précisément, il existe un seul élément de A qui soit le plus proche de B, et réciproquement. Dans la figure 1.6, par exemple, le sommet S_A est l'élément de A le plus proche de l'objet B, et l'arête A_B est l'élément de B le plus proche de l'objet A. Plusieurs algorithmes utilisent cette propriété pour détecter des interpénétrations entre objets polyédriques convexes ou même calculer la distance qui les sépare. Ainsi, Gilbert *et al.* [GJK88] présentent une méthode fondée sur les différences de Minkovski et l'optimisation convexe pour calculer la distance entre objets convexes en déterminant les éléments les plus proches. Aussi, l'algorithme de Lin et Canny [LC91] utilise la cohérence spatiale et temporelle de la scène en remarquant que, d'un instant à l'autre, les objets ne se déplacent pas beaucoup, et que les plus proches éléments pour un instant donné t_n peuvent être cherchés, en temps espéré constant, dans le voisinage des plus proches éléments pour l'instant d'avant t_{n-1}. La revendication du temps constant n'est cependant justifiable que lorsque la quantité de rotation de l'objet d'un instant à l'autre est faible. Il a ainsi été montré expérimentalement que le temps passé à chercher les nouveaux plus proches éléments dépend à peu près linéairement de cette quantité de rotation [Zac00]. Afin d'éviter que la recherche des nouveaux plus proches éléments ne prenne trop de temps lorsque les objets se sont beaucoup déplacés entre t_{n-1} et t_n, et que les nouveaux plus proches éléments sont très éloignés des anciens, l'algorithme de Lin et Canny [LC91] a été étendu aux représentations hiérarchiques d'un polyèdre, introduites par Dobkin et Kirkpatrick [DK85b], par Guibas *et al.* [GHZ99]. La méthode consiste à passer par les représentations moins détaillées du polyèdre pour accélérer le déplacement à la surface du polyèdre, lors de la recherche des nouveaux plus proches éléments.

L'idée d'utiliser la cohérence temporelle s'avère ainsi extrêmement puissante et permet fréquemment d'accélérer la détection de collisions de façon très significative [Bar90, LC91, Lin93, LC94, PML97, LC98].

Cette idée se retrouve également dans une méthode efficace permettant de résoudre le *problème des n objets*. Lorsque n objets peuvent se déplacer en même temps, il est a priori nécessaire d'effectuer $O(n^2)$ détections de collisions. Afin d'éviter ce problème, Cohen *et al.* [CLMP95] utilisent une technique de réduction qui consiste à englober chaque objet dans une boîte parallèle aux axes et à remarquer que lorsque deux boîtes englobantes se recouvrent, leurs projections suivant les trois axes se re-

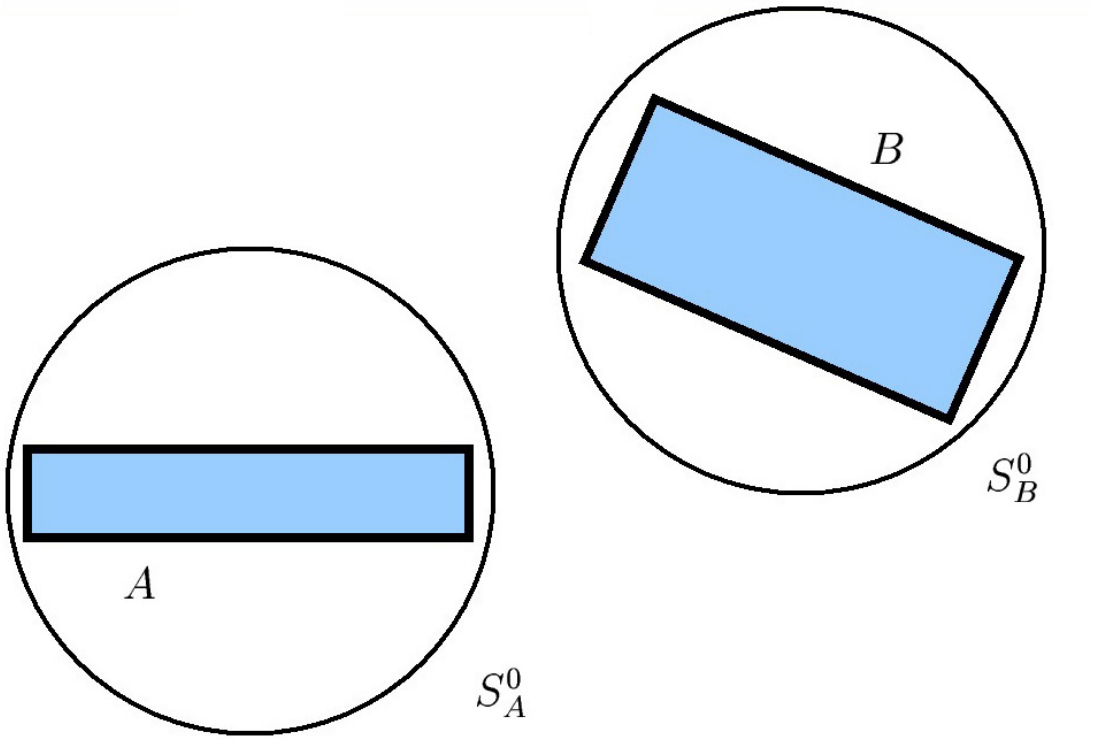


Fig. 1.7 – Les deux sphères englobantes ne se recouvrent pas. Les deux objets ne peuvent donc pas s'interpénétrer.

couvrent également. La méthode consiste donc à maintenir une liste triée des intervalles représentant les projections des boîtes et à parcourir ces listes pour détecter des recouvrements entre intervalles. La clé du succès de cette méthode repose ici aussi sur la cohérence temporelle et spatiale de l'environnement virtuel : en un pas de temps, les objets se déplacent peu, et les listes d'intervalles pour un instant donné t_n sont presque les mêmes à l'instant suivant t_{n+1}. Par conséquent, le tri (et le parcours) de ces listes prend un temps espéré linéaire.

Notons que, depuis peu, plusieurs méthodes utilisent les cartes graphiques pour détecter des interpénétrations entre objets. Brièvement, le principe est de projeter les objets suivant plusieurs directions (ce qui revient à les observer sous plusieurs points de vue) et à déterminer les intersections entre les projections [SF91, BWS99] (ce principe de projection, ou de réduction de dimension, est le même que celui employé dans les algorithmes résolvant le problème à n objets [CLMP95]). Dans d'autres cas, la carte graphique est utilisée dans certaines parties de l'algorithme seulement. C'est le cas par exemple de la méthode proposée par Kim *et al.* pour déterminer la quantité d'interpénétration entre deux objets polyédriques quelconques [KOLM02]. Dans cet algorithme, la carte graphique est utilisée pour déterminer le point le plus proche de l'origine dans une différence de Minkowski de deux polyèdres, qui est aussi un polyèdre, en la visualisant depuis l'origine. Le z-buffer de la carte graphique est alors

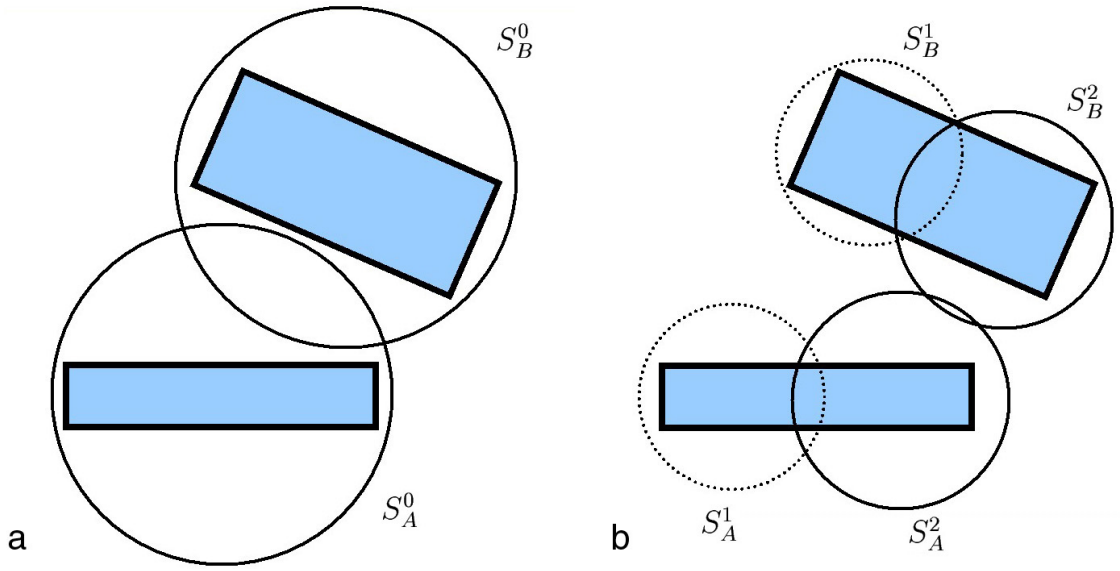


Fig. 1.8 – a : les deux sphères englobantes se recouvrent. b : des sphères englobantes plus petites permettent de mieux localiser l'éventuelle zone de contact.

utilisé pour déterminer, à la précision de la carte graphique, le point le plus proche de l'origine. Lombardo *et al.* [LCN99], également, utilisent la carte graphique pour détecter des collisions entre un outil chirurgical et un foie virtuels. La méthode consiste à considérer l'outil chirurgical, ou une approximation de sa trajectoire (ce qui en fait une méthode pseudo-continue), comme la zone de visualisation du foie virtuel : une collision intervient entre l'outil et le foie si et seulement si ce dernier apparaît dans cette zone de visualisation.

Une stratégie extrêmement répandue en détection de collisions (ainsi que dans d'autres domaines, par exemple en lancer de rayons) consiste à utiliser des hiérarchies de volumes englobants pour accélérer la détection. En quelques mots, des tests de recouvrement entre volumes englobants sont utilisés pour éliminer simplement de nombreux tests élémentaires non pertinents. Considérons en effet deux objets polyédriques A et B composés chacun de n_A et n_B triangles, respectivement. Il faudrait en théorie effectuer un test d'interpénétration pour chacune des paires de triangles possibles, soit $n_A n_B$ tests. Ceci rendrait la détection d'interpénétrations impraticable pour des objets complexes. Afin d'éviter ce problème (qui n'est bien sûr pas sans rapport avec le problème des n objets), chaque objet est associé à une structure englobante, organisée en arbre, qui permet de localiser rapidement les zones de contact éventuelles et d'éliminer rapidement les parties des objets éloignées les unes des autres.

Supposons par exemple que chacun des deux objets A et B en train d'être examinés par les fonctions de détection de collisions soient englobés dans une sphère, S_A^0 et S_B^0 respectivement. Si les deux sphères ne se recouvrent pas, alors il ne peut pas y avoir d'interpénétrations entre les objets (figure 1.7). Si, par contre, les sphères se recouvrent (figure 1.8.a), alors il *peut* y avoir une interpénétration entre les objets (mais

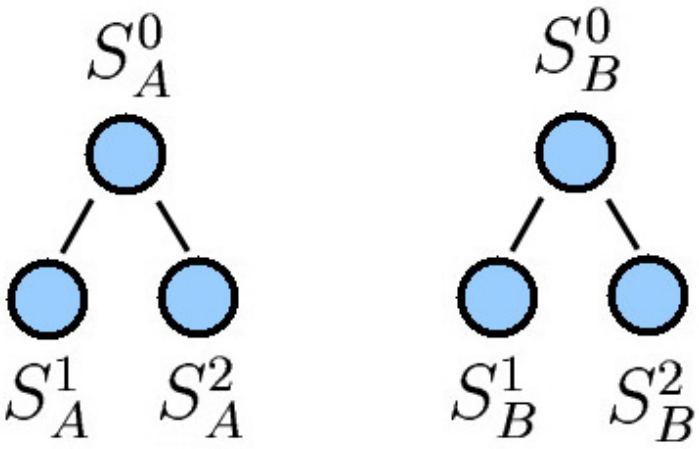


Fig. 1.9 – Les hiérarchies englobantes des deux objets A et B.

ce n'est pas obligatoire puisque les sphères ne correspondent pas exactement aux objets englobés). Chaque sphère est alors remplacée par plusieurs sphères plus petites, dont la réunion recouvre l'objet. Les tests de recouvrement sont alors effectués entre les paires de sphères plus petites. Dans l'exemple de la figure 1.8.b, ces sphères sont S_A^1 et S_A^2 pour l'objet A, et S_B^1 et S_B^2 pour l'objet B. Le recouvrement entre les sphères S_A^2 et S_B^2 permet ainsi de mieux localiser l'éventuelle collision.

Ce processus est effectué récursivement jusqu'à ce que les sphères soient plus petites qu'un seuil prédéterminé. Les tests d'interpénétration sont alors effectués sur les morceaux d'objet eux-mêmes (par exemple des tests d'interpénétration entre triangles, pour des objets polyédriques). De manière générale, les volumes englobants sont ainsi naturellement organisés en hiérarchies. Les hiérarchies de l'exemple précédent sont données dans la figure 1.9. Les noeuds internes de la hiérarchies sont les volumes englobants. Les noeuds S_A^0 et S_B^0 sont les noeuds *racines* de la hiérarchies, et les noeuds S_A^1 et S_A^2 (respectivement S_B^1 et S_B^2) sont les *fils* de S_A^0 (respectivement S_B^0). Les noeuds *feuilles*, c'est-à-dire les noeuds qui n'ont pas de descendance, sont aussi des volumes englobants, mais qui sont associés à la partie de la géométrie de l'objet qu'ils contiennent.

Les volumes englobants typiquement utilisés en détection de collisions sont les sphères [Qui94, Hub95, RKK97, BS02], les boîtes englobantes isothétiques (*i.e.* dont les axes sont parallèles aux axes du repère global) [VDB98], les boîtes englobantes orientées [Got99, GLM96], et les k-dops [KHMSZ98]. D'autres types de volumes englobants, plus exotiques, sont les coquilles sphériques [KPLM98] et les arbres QuOSPO [He99].

Les quatre premiers types de volumes englobants sont visibles dans la figure 1.10. Les k-dops sont des *polytopes à orientation discrète*. Essentiellement, les orientations des faces du polytope englobant sont choisies parmi un petit nombre d'orientations fixées. Il est possible de conclure à une séparation de deux k-dops dont les orientations appartiennent au même ensemble lorsque les projections des deux k-dops sont disjointes suivant chaque orientation. Ceci n'est cependant nécessaire que lorsque l'en-

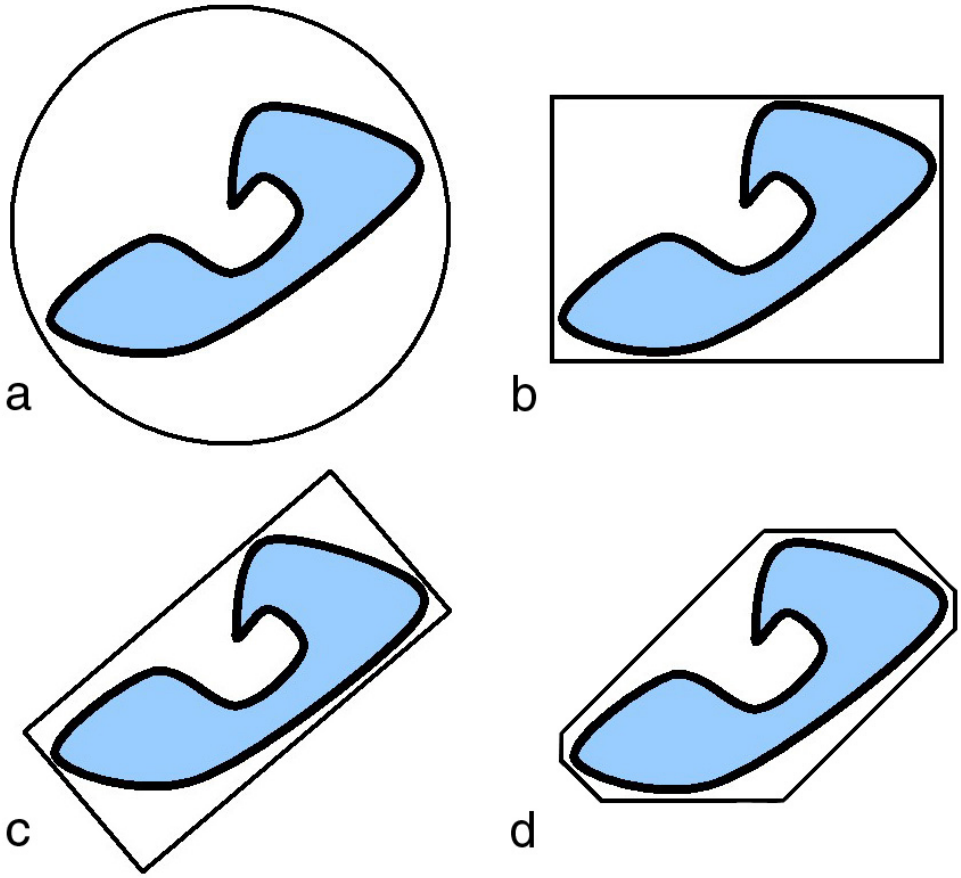


Fig. 1.10 – Quatre types fréquents de volumes englobants. a : sphère. b : boîte englobante isothétique. c : boîte englobante orientée. d : 4-dop (en deux dimensions).

semble d'orientations est *fermé* [Klo98]. Dans le cas contraire, les deux k-dops peuvent être disjoints même lorsque leurs projections se recouvrent suivant chacune des orientations[1].

Nous verrons cependant au chapitre 3 qu'employer de tels tests *conservatifs*, qui assurent qu'un recouvrement est toujours détecté, mais déclarent parfois *à tort* que des volumes englobants se recouvrent, peut être toutefois suffisament efficace pour accélérer la détection de collisions de façon significative.

Notons que, pour des objets rigides, il est souvent possible de calculer ces hiérarchies avant le début de la simulation. C'est le cas lorsque les volumes englobants utilisés sont des sphères ou des boîtes englobantes orientées. Lorsque des boîtes isothétiques ou des k-dops sont employés, il est nécessaire de mettre à jour les volumes englobants

[1]Dans [Klo98], la fermeture d'un ensemble de vecteurs est l'ensemble des produits vectoriels des arêtes du k-dop défini à partir de cet ensemble de vecteurs. Un ensemble de vecteurs est alors dit fermé lorsqu'il est égal à sa fermeture. Il est possible de montrer que le test de séparation de deux k-dops est exact si et seulement si l'ensemble des vecteurs définissant les orientations des faces des k-dops est fermé.

quand les objets subissent une rotation [VDB98, Zac00].

Généralement, il est d'autant plus difficile de faire un test de recouvrement entre deux volumes englobants de même type que ceux ci sont mieux adaptés à la géométrie de l'objet. Par exemple, il est très simple de savoir si deux sphères se recouvrent, en comparant la distance séparant leurs centres à la somme de leurs rayons. Cependant, il est plus difficile de déterminer si deux boîtes englobantes orientées se recouvrent, et la version la plus efficace requiert en moyenne deux cents opérations élémentaires (additions et autres) [GLM96].

Grâce à leur organisation en hiérarchies, il est possible d'écrire simplement l'algorithme de parcours récursif des volumes englobants. En supposant par exemple que les objets A et B sont des ensembles de triangles, et que les noeuds feuilles de la hiérarchie ne contiennent chacun qu'un seul triangle, l'algorithme peut être décrit de la manière suivante :

DétecterRecouvrement(noeud N_A, noeud N_B)

1. Si les noeuds N_A et N_B ne se recouvrent pas, FIN.
2. Si N_A et N_B ne sont pas des feuilles, exécuter DétecterRecouvrement(F_A, F_B) pour tous les fils F_A de N_A et tous les fils F_B de N_B.
3. Si N_A est une feuille et N_B n'est pas une feuille, exécuter DétecterRecouvrement(N_A, F_B) pour tous les fils F_B de N_B.
4. Si N_A n'est pas une feuille et N_B est une feuille, exécuter DétecterRecouvrement(F_A, N_B) pour tous les fils F_A de N_A.
5. Si N_A et N_B sont des feuilles, détecter une interpénétration entre les triangles contenus dans les feuilles. Le cas échéant, ajouter la paire de triangles à la liste de triangles en collision et retourner cette nouvelle liste. Si les triangles ne s'interpénétrent pas, retourner la liste inchangée.

Cet algorithme peut être sujet à des variantes, notamment lorsque les deux noeuds sont internes (étape 2). Ainsi, il peut par exemple être choisi de parcourir d'abord la descendance du noeud de plus grand volume [Got99]. Il peut être aussi choisi d'arrêter la détection de collisions dès que deux triangles qui s'interpénètrent ont été trouvés, au lieu de recenser toutes les paires de triangles en collision (étape 5).

Notons enfin que d'autres types de structures hiérarchiques peuvent être utilisées. Ces structures ne sont pas fondées sur des hiérarchies de volumes englobants mais sur des subdivisions récursives de l'espace de l'objet. Les méthodes les plus connues sont les arbres BSP (*binary space partitioning*) [NAT90], qui utilisent des plans d'orientation quelconque, les *kd*-trees et les octrees [Sam89]. Ces structures, cependant, sont efficaces lorsqu'il s'agit d'*éliminer* un grand nombre de tests d'un seul coup, mais perdent en efficacité à mesure que les objets se rapprochent les uns des autres et qu'un grand nombre de tests de recouvrement doit être effectué. Ils sont ainsi souvent délaissés pour des hiérarchies de volumes englobants [Lin97].

Dans ce mémoire, nous allons proposer des algorithmes de détection de collisions continue accélérés grâce à des hiérarchies de volumes englobants. Pour les algorithmes

discrets, l'équation traditionnelle du coût d'une détection de collisions discrète entre deux objets est [GLM96] :

$$T_D = C_{BV} \times N_{BV} + C_T \times N_T \tag{1.5}$$

où T est le temps total pour effectuer une détection de collisions, C_{BV} (respectivement C_T) est le temps requis pour effectuer un test de recouvrement entre volumes englobants (respectivement entre triangles), et où N_{BV} (respectivement N_T) est le nombre de tests de recouvrements entre volumes englobants (respectivement entre triangles).

Dans le cas de la détection de collisions continue entre objets polyédriques, cette équation peut être détaillée pour faire intervenir les tests arête/arête, point/face et face/point. Cependant, en supposant que les objets ne comportent pas de faces dégénérées, le coût de la détection de collisions entre deux triangles, qui comporte un nombre constant de tests AA, PF et FP, est bien représenté par une équation du type (1.5).

1.3.2 détection de collisions continue

Il existe relativement peu de méthodes de détection de collisions continues. Canny [Can86] utilise une paramétrisation des trajectoires d'objets polyédriques basée sur des quaternions afin de calculer les instants de collisions en résolvant des polynômes de faible degré. Dans le cas d'un seul objet mobile dans un environnement statique, la méthode conduit à un polynôme de troisième degré. Cependant, l'absence de prétraitement permettant d'éliminer rapidement les tests élémentaires non pertinents rend la méthode trop complexe pour pouvoir interagir avec des objets en temps réel.

Cameron [Cam90] présente une méthode d'extrusion spatio-temporelle pour déterminer des contacts entre des objets CSG (*Constructive Solid Geometry*, géométrie solide constructive) mobiles. En raison du coût important de l'opération d'extrusion, cependant, les mouvements des objets sont restreints à des translations par morceaux. Un autre travail portant sur les objets CSG est celui de Fedrowitz [FD98], qui étend la méthode des bornes spatiales (*S-bounds*) introduite par Cameron [Cam89] pour détecter des recouvrements de trajectoires d'objets plus efficacement que l'extension proposée dans [Cam90]. La méthode consiste à utiliser non plus des boîtes englobantes alignées sur les axes, mais des groupes de telles boîtes englobantes, intérieures ou extérieures aux objets, pour mieux approcher la géométrie des objets et pouvoir utiliser l'algèbre sur ces structures englobantes introduite par Sanna et Montuschi [SM95].

Von Herzen *et al.* [VHBZ90] utilisent des bornes de Lipschitz et une méthode de dichotomie pour trouver l'instant de premier contact entre des surfaces paramétrées dépendant du temps. Des hiérarchies de sphères englobantes et de boîtes englobantes isothétiques sont calculées pendant la détection de collision pour accélérer la méthode.

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.* [SWFCB93] utilisent l'arithmétique d'intervalles et des méthodes de Newton par intervalles pour accélérer significativement ces approches, et emploient un test simple d'élimination par sphères englobantes (une seule sphère par objet) pour résoudre le problème de la détection de collisions entre plus de deux objets (*le problème des n objets*). Bien que l'algorithme présente de nombreuses caractéristiques intéressantes et soit utilisable pour des animations, il n'est pas assez efficace pour permettre l'interaction en temps réel, même pour des objets rigides [Sny95]. Des objets polyédriques quelconques peuvent être considérés comme des unions de surfaces paramétrées simples (sommets, arêtes et faces). Cependant, aucune des méthodes fondées sur l'arithmétique d'intervalles pour détecter des collisions entre surfaces paramétrées n'utilisent des volumes englobants pour accélérer la détection entre des *unions* d'objets. Par conséquent, détecter une collision entre deux objets polyédriques quelconques est transformé artificiellement en un problème à n objets (voir par exemple l'algorithme pour éléments multiples de Snyder *et al.* [SWFCB93]).

Mirtich [Mir96] n'utilise que des mouvements ballistiques 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 (notamment lorsque le simulateur autorise les contacts transitoires non instantanés, lorsque les objets peuvent glisser les uns sur les autres[2]), et le pas de temps variable peut ne pas être adapté aux applications interactives.

Notons que certaines méthodes effectuent de la détection de collisions discrète sur les hiérarchies de volumes englobants, et de la détection de collisions continue sur les primitives polyédriques (*i.e.* les sommets, les arêtes et les faces) [LSW99]. D'autres méthodes englobent dans un même volume l'objet (ou ses volumes englobants) aux instants initiaux et finaux [ES99]. Cependant, ces méthodes ne garantissent pas que la trajectoire *complète* de l'objet est englobée, et peuvent ainsi manquer des collisions. Le seul cas où englober la position initiale et finale peut être considéré comme valide est celui, beaucoup plus simple, ou l'un des objets est un simple point mobile. Dans ce cas, certaines méthodes parviennent à détecter des collisions en continu à des taux de rafraîchissement suffisamment élevés pour l'haptique [GLGT00].

Enfin, insistons sur le fait que les hiérarchies de volumes englobants sont encore utilisables en détection de collisions continue. Le problème est cependant maintenant de détecter des recouvrements entre volumes englobants *au cours d'un intervalle de temps* (le calcul de l'instant de premier contact entre les volumes englobants n'est pas nécessaire) pour, le cas échéant, détecter des collisions (*i.e.* l'instant de premier contact) entre les primitives contenues entre les feuilles de la hiérarchie englobante.

[2]Ceci n'arrive pas dans un simulateur par impulsions, où les objets rigides non articulés n'ont toujours que des trajectoires ballistiques [MC95]. Un tel simulateur, cependant, peut difficilement prendre en compte les empilements d'objets [Mir96].

1.3.3 Discussion

Les méthodes de détection de collisions discrètes, bien que plus simples à mettre en oeuvre et très fréquemment utilisées dans les simulateurs dynamiques, présentent de sérieux inconvénients. Outre le manque de réalisme physique résultant de la pénétration, ces méthodes peuvent manquer des collisions lorsque les objets sont trop fins ou trop rapides. Bien qu'un pas de temps adaptatif et des méthodes prédictives puissent être utilisés pour corriger ce problème dans des applications *offline*, ceci peut parfois ne pas être adapté dans des applications interactives lorsqu'un taux de rafraîchissement d'image relativement élevé et constant est requis.

Un second problème lié à la discrétisation des trajectoires est que des méthodes de *backtracking* doivent être employées pour calculer l'instant de premier contact, souvent nécessaire dans les méthodes de simulation dynamique par contraintes [FMM77, MW88, Bar90]. Ces méthodes consistent à rechercher l'instant de premier contact par récursion. Supposons que l'intervalle de temps courant soit $[t_n, t_{n+1}]$. Essentiellement, un instant de premier contact t_e est *estimé* dans cet intervalle (par exemple en prenant l'instant intermédiaire $\frac{t_n+t_{n+1}}{2}$, ou par une méthode d'interpolation linéaire ou quadratique dépendant des vitesses et accélérations des objets). Les positions des objets sont alors calculées à cet instant et une détection d'interpénétration est de nouveau effectuée. Selon que les objets s'interpénètrent ou pas, l'algorithme en déduit que le premier instant de collision est dans $[t_n, t_e]$ ou $[t_e, t_{n+1}]$, respectivement, et boucle sur ce nouvel intervalle. Le processus s'arrête lorsque la quantité d'interpénétration est inférieure à un certain seuil.

De telles méthodes de backtracking peuvent avoir un coût élevé (et difficilement prévisible) lorsque les objets sont complexes ou qu'ils se sont fortement interpénétrés. De plus, le backtracking n'étant enclenché que quand une interpénétration a été détectée, des objets non connexes, ou même simplement non convexes, peuvent se retrouver dans une configuration dont ils ne pourraient sortir (c'est le cas par exemple de deux cerceaux qui à un instant seraient entremêlés alors qu'ils ne l'étaient pas à l'instant d'avant). De tels problèmes sont rencontrés par exemple dans la technique d'interaction décrite dans Snyder [Sny95]. Notons que certaines méthodes effectuent un backtracking approché [FW99] : elles estiment la position et l'instant du premier contact en considérant que la trajectoire de chaque sommet de l'objet polyédrique est un segment de droite. Ceci n'est bien sûr qu'une approximation lorsque l'objet à un mouvement de rotation, puisque cela conduit à une déformation de l'objet au cours de l'intervalle de temps.

Un moyen d'éviter d'avoir recours à des méthodes de backtracking est d'autoriser les interpénétrations entre objets, ce qui peut être difficile à justifier physiquement, et à déterminer la quantité d'interpénétration. Ce problème est cependant extrêmement difficile pour des objets généraux (non convexes), et les meilleurs résultats actuels ne prennent pas en compte la trajectoire de l'objet pour déterminer cette quantité d'interpénétration [KOLM02]. Ainsi, dans le cas représenté dans la figure 1.11, le point mobile est à l'intérieur de l'obstacle à l'instant t_2, mais à cet instant la plus petite

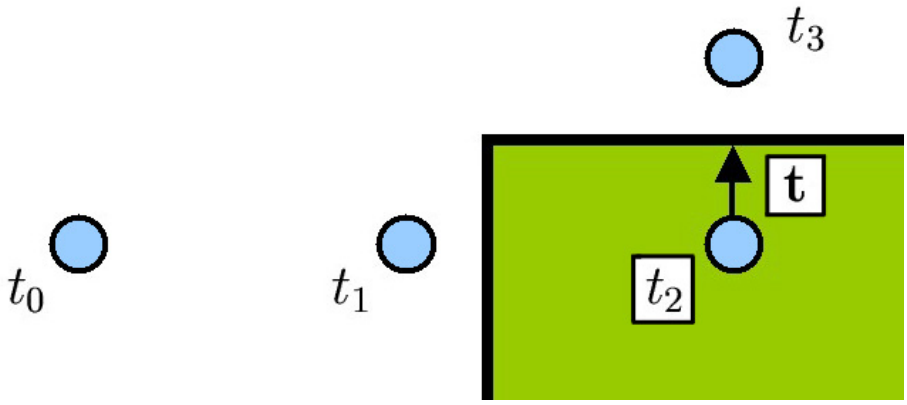


Fig. 1.11 – Autoriser l'interpénétration des objets peut conduire à des incohérences dans la simulation. Dans cet exemple, l'objet qui a pénétré à l'instant t_2 ressort du mauvais côté de l'obstacle.

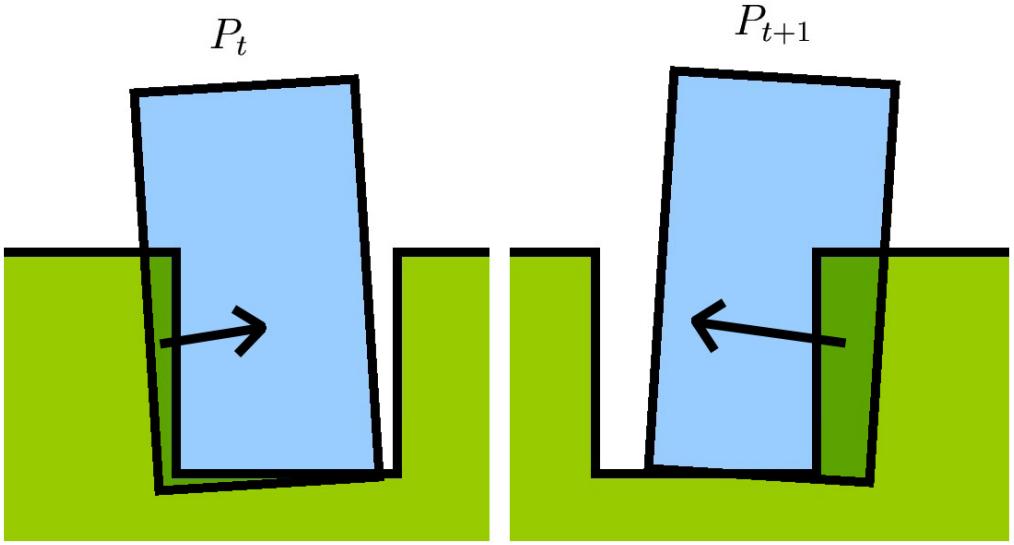


Fig. 1.12 – Les interpénétrations entre objets peuvent rendre la simulation instable.

interpénétration est représentée par le vecteur $\mathbf{t}$, ce qui amène à sortir le point mobile par le haut de l'obstacle en l'instant t_3. Notons toutefois que la quantité d'interpénétration peut être utilisée pour accélérer les méthodes de backtracking, en conduisant à une meilleure estimation de l'instant de premier contact t_e.

Enfin, l'interpénétration des objets peut être une cause majeure d'instabilité

des simulations dynamiques. Ce qui ne prête pas à conséquence dans une simulation purement virtuelle (sans intervention de l'utilisateur) a beaucoup plus d'importance lorsque un périphérique haptique est utilisé pour interagir avec l'environnement virtuel. Si l'on se réfère par exemple à la configuration classique représentée dans la figure 1.12, destinée à tester des méthodes d'interaction, de deux objets imbriqués (*peg-in-a-hole*), il est clair qu'une première interpénétration de l'objet d'un côté de l'orifice à un instant t (position P_t) entraîne une force importante pour supprimer l'interpénétration. Cette force conduit souvent à une pénétration plus importante du côté opposé de l'orifice à l'instant suivant $t+1$ (position P_{t+1}), qui crée elle-même une force de réaction plus importante que la précédente. Cette oscillation, en s'amplifiant, entraîne l'instabilité de la simulation [GMELM00].

1.4 Méthodes de simulation dynamique

Il existe également une littérature abondante sur les méthodes de simulation dynamique. Cependant, de même qu'en détection de collisions il est possible d'établir une distinction entre les approches plutôt théoriques provenant de la géométrie algorithmique, et les approches plutôt pratiques destinées à l'infographie et la réalité virtuelle, de même en simulation dynamique il est possible de grossièrement répartir (historiquement au moins) les approches de simulation dynamique en deux grandes catégories, selon qu'elles proviennent des mathématiques appliquées ou de l'infographie. Une excellente introduction aux méthodes provenant des mathématiques appliquées est donnée dans le mémoire de thèse de Franck Génot [Gen98]. Les méthodes provenant de l'infographie sont exposées par exemple dans [Bar93] et [Zac00].

Essentiellement, les méthodes mathématiques s'attachent à définir des schémas d'intégration et des modèles physiques rigoureux. Nous avons ainsi remarqué précédemment que les lois physiques doivent être *discrétisées* pour être résolues. Le schéma d'intégration choisi pour résoudre ces équations discrétisées, et même la façon de discrétiser les équations, a une grande importance dans la qualité de la solution obtenue [WB97]. Aussi, une partie des travaux s'attache à formuler des schémas d'intégration précis et efficaces [Lot81, Lot84, MP88]. Il est possible de préciser ces deux derniers termes en quelques mots :

- la *précision* concerne la différence entre l'approximation de la solution, obtenue en discrétisant l'équation différentielle, et la solution exacte ;
- l'*efficacité* concerne à la fois le coût du passage d'un instant discret à l'instant discret suivant, et le pas de temps utilisé par le schéma d'intégration. Certaines méthodes d'intégration, le schéma d'Euler explicite par exemple, sont peu coûteuses *par pas de temps*, mais sont parfois obligées d'utiliser un très petit pas de temps lorsque les équations sont *raides*, afin d'éviter une divergence de la solution [WB97].

Pour plus de détails sur les schémas d'intégration des équations différentielles ordinaires, nous renvoyons le lecteur à [WB97] ou [PTVF92], par exemple.

La plupart de ces méthodes formulent les problèmes dynamiques comme un problème de complémentarité linéaire (LCP), qui relie les forces et les accélérations [Lot84, Bar94, ST96, AP97, APS99]. Par exemple, dans le problème du mouvement contraint sans friction, le LCP exprime la relation entre les forces de contact et les accélérations normales relatives aux points de contact. Si $\mathbf{f}$ et $\mathbf{a}_{cp}$ sont deux vecteurs dans $\mathbb{R}^m$ décrivant les composantes normales des forces de contact[3] et les composantes normales des accélérations relatives aux m points de contact, alors il existe une matrice $\mathbf{A}$ de taille $m \times m$ et un vecteur $\mathbf{b}$ de $\mathbb{R}^m$ tels que :

$$\mathbf{a}_{cp} = \mathbf{A}\mathbf{f} + \mathbf{b}, \;\; \mathbf{a}_{cp} \geqslant 0, \;\; \mathbf{f} \geqslant 0, \;\; \mathbf{a}_{cp}^T \mathbf{f} = 0 \tag{1.6}$$

La relation de complémentarité $\mathbf{a}_{cp}^T \mathbf{f} = 0$ exprime le fait que, pour chaque point de contact, soit l'accélération normale relative est non nulle (le point de contact disparait) et alors la force de contact est nulle, soit la force de contact est non nulle (les objets restent en contact) et donc l'accélération normale relative est nulle.

La difficulté de choisir un schéma d'intégration possédant de bonnes propriétés est étroitement reliée à la difficulté de définir un modèle physique réaliste, et possédant de bonnes propriétés mathématiques. Ainsi, alors que les lois physiques des systèmes sans frottements sont parfaitement connues, et possèdent de bonnes propriétés mathématiques, les problèmes dynamiques sont beaucoup plus complexes dès lors que la friction est incluse dans les modèles, et sont parfois *inconsistants* (*i.e.* ont plus d'une solution) ou *indéterminés* (*i.e.* n'ont pas de solutions). Il en est ainsi lorsque le modèle de frottements de Coulomb est utilisé, comme illustré par le célèbre exemple de Painlevé, qui considère une barre rigide en contact avec le sol [Pai1895, Bar91, Gen98]. Des travaux récents, cependant, montrent que lorsque le système présenté par Painlevé est convenablement initialisé (c'est-à-dire dans un état consistant), les problèmes dynamiques ont toujours une solution (au prix d'une solution impulsive) [Gen98, GB98]. Notons, toutefois, que le réalisme même des formulations LCP est discuté par certains auteurs [Cha99].

A l'opposé, les premières méthodes apparues en infographie ont privilégié la vitesse des algorithmes plutôt que leur rigueur physique, et de nombreuses approches ont utilisé la méthode des *pénalités* [Moo87, MW88, Cun88, KZ90, Wil87]. Brièvement, la méthode des pénalités consiste à appliquer à l'objet une force proportionnelle à la quantité d'interpénétration (comme si un ressort empêchait les objets de s'interpénétrer localement). Les méthodes de pénalité sont extrêmement simples à implanter (lorsque la quantité d'interpénétration est connue), mais difficiles à justifier physiquement. Notamment, un problème des méthodes de pénalité vient de ce qu'elles sont seulement locales. Ainsi, dans le cas d'une table posée sur le sol, la force totale exercée par le sol, pour une pénétration donnée, dépend du nombre de pieds de la table, ce qui n'est évidemment pas réaliste, et rend le choix des constantes de raideur des ressorts difficile. Par ailleurs, il est souvent nécessaire d'utiliser des constantes de raideur élevées pour que les objets ne s'interpénètrent pas trop, ce qui conduit à des équations différentielles

[3] Dans un système sans frottements, les composantes tangentielles des forces de contact sont nulles.

raides qui peuvent conduire à des instabilités dans la simulation [Bar89, WB97]. Notons que Barzel *et al.* [BB88] présentent une méthode de dynamique *inverse* appliquée à la modélisation : pour une accélération donnée, déterminée en fonction de contraintes bilatérales à respecter, les forces exercées sur l'objet sont calculées en résolvant un système linéaire. Cette méthode s'apparente à celle des pénalités, mais n'est pas locale puisque toutes les forces sont calculées simultanément. Notons aussi que des travaux récents ont proposé des méthodes pour réduire les problèmes principaux des méthodes par pénalité, à savoir le choix des constantes de raideur et le choix du pas de temps [JDL98].

Sans doute en raisons des faiblesses des méthodes de pénalité, et grâce à l'amélioration des performances du matériel, des techniques de plus en plus rigoureuses sont apparues en infographie, réduisant la distinction originelle entre les approches mathématiques et infographiques. Ce phénomène peut être notamment illustré par les travaux de David Baraff. Ainsi, Baraff [Bar89] présente une méthode analytique pour prendre en compte les contraintes unilatérales de non-pénétration dans la simulation d'objets rigides. Les problèmes dynamiques sans friction sont formulés comme un problème quadratique mais une heuristique est utilisée pour éviter de le résoudre. L'heuristique consiste à essayer de déterminer à l'ensemble des points de contact qui vont disparaître à l'instant suivant. La méthode est étendue aux surfaces courbes rigides dans [Bar90], mais le problème quadratique est cette fois résolu par l'algorithme de Lemke [Tom76]. Le problème du frottement est étudié dans [Bar91]. Les problèmes dynamiques sont formulés comme un problème de complémentarité linéaire (LCP) dans [Bar94]. Dans le cas sans frottements, l'algorithme résolvant les problèmes dynamiques est équivalent à l'algorithme de Dantzig résolvant les LCPs [CD68]. L'algorithme résolvant le LCP est ensuite adapté pour prendre les frottements de Coulomb en compte dans le problème du mouvement contraint. Ruspini *et al.* ont étendu cette approche aux formulations employant les coordonnées généralisées dans [RK97]. Kawachi [Kaw98] étend la formulation de Baraff [Bar94] pour inclure les frottements dans le problème de la réponse à la collision.

Notons que quelques méthodes, appliquées à l'infographie ou la réalité virtuelle, formulent les problèmes dynamiques comme des problèmes de complémentarité non-linéaire (NCP) [BS98, SS98a]. Ces problèmes sont ensuite résolus par une méthode du point fixe, en linéarisant les problèmes NCP.

Notons aussi que les contraintes bilatérales sont plus faciles à prendre en compte que les contraintes unilatérales. Ainsi, un problème dynamique ne comportant que des contraintes bilatérales peut être résolu en calculant la solution d'un système linéaire. Lorsqu'un système comporte des contraintes unilatérales, il est nécessaire de déterminer celles qui vont persister. Ce problème est un problème combinatoire, ce qui accroît considérablement la difficulté de la résolution. Ainsi, Baraff [Bar96] présente une méthode en temps linéaire pour calculer la dynamique d'un système articulé ne comportant pas de boucles. La méthode repose sur la factorisation de la jacobienne décrivant le système. Cette jacobienne est creuse et posséde un ordre d'élimination parfait lorsque le système est acyclique (*i.e.* ne comporte pas de boucles), et peut être

factorisée en temps linéaire. Pour gérer les boucles et les contraintes unilatérales, cependant, l'auteur relie cette méthode à ses travaux précédents [Bar94], ce qui conduit à une complexité polynomiale (en $O(l^3)$, où l est le nombre de boucles et de contraintes unilatérales). La méthode à été améliorée par Faure pour mieux prendre en compte les boucles [Fau99]. Le principe consiste à résoudre d'abord les contraintes acycliques, puis à utiliser une méthode d'optimisation contrainte pour les boucles. Cette méthode permet d'obtenir un temps quadratique en le nombre de boucles.

Une autre approche a été proposée par Mirtich et Canny [MC95, Mir96], qui suggèrent de résoudre tous les types de contraintes par des impulsions. Par exemple, les contraintes de non-pénétration sont résolues par des successions de micro-collisions entre objets en contact. Les auteurs, cependant, reconnaissent que cette approche peut être mal adaptée lorsque de nombreux objets sont empilés, et suggèrent d'utiliser une méthode hybride, par impulsions et par contraintes.

Récemment, Milenkovic [MS01] a proposé une méthode pour résoudre, dans la pratique, les problèmes posés par les empilements d'objets. Lorsque de nombreux objets sont empilés les uns sur les autres, de nombreuses collisions interviennent et le simulateur doit fréquemment résoudre des problèmes de réponse à la collision et ne peut pas utiliser un grand pas de temps pour faire progresser la simulation. La méthode proposée consiste à stabiliser les empilements d'objets en *synchronisant*, pour un pas de temps donné, toutes les collisions qui pourraient intervenir durant le pas de temps. Ainsi, chaque pas de temps comporte une phase de mise à jour des positions des objets (pour synchroniser les impacts), puis des vitesses (pour résoudre les impacts), puis des accélérations (pour calculer les mouvements contraints). Essentiellement, c'est la phase de mise à jour des positions de l'algorithme représenté dans la figure 1.1 qui est adaptée. Bien que cette méthode perde en rigueur physique, elle permet de conserver un réalisme suffisant pour l'infographie et, surtout, résoud convenablement le problème des empilements d'objets.

1.5 Contributions

Nous avons pu constater, dans la section 1.3, qu'il existe un besoin important de méthodes de détection de collisions continues. Les méthodes discrètes, en effet, présentent plusieurs inconvénients importants, notamment lorsque le simulateur doit être utilisé dans une application interactive, éventuellement avec retour d'efforts, en raison des risques d'instabilité. Par ailleurs, nous pensons avoir montré que les méthodes continues actuelles ne sont pas assez efficaces pour la détection de collisions entre objet polyédriques [Can86, Cam90, SWFCB93], et que certaines d'entre elles, utilisées dans des simulateurs interactifs, ne sont pas exactement continues puisqu'elles n'englobent pas complètement la trajectoire de l'objet au cours du mouvement [LSW99, ES99].

La première partie de ce mémoire va s'attacher à proposer une méthodologie pour détecter des collisions entre objets polyédriques en continu. Essentiellement, dans la

première partie du chapitre 2, nous allons introduire et justifier l'utilisation de *mouvements intermédiaires arbitraires* pour interpoler les positions successives des objets rigides et détecter des collisions en continu efficacement. Dans une première approche, décrite dans la suite du chapitre 2, nous allons chercher à définir un mouvement intermédiaire particulier, permettant de résoudre efficacement les équations de détection de collisions (1.3) et (1.4). Nous allons ainsi montrer que ces équations peuvent être réduites à des polynômes de degré trois, dont les racines peuvent être calculées efficacement. Nous allons également montrer que ce même mouvement intermédiaire peut être utilisé pour détecter des recouvrements en continu entre sphères englobantes aussi efficacement (c'est-à-dire en résolvant aussi des équations polynomiales de troisième degré), qui seront exploitées pour accélérer la détection de collisions. Ces résultats ont été exposés dans [RKC00] et [RKC01].

Dans le chapitre 3, nous allons explorer une deuxième approche, fondée elle aussi sur l'utilisation de mouvements intermédiaires arbitraires, qui va consister non plus à réduire le *coût* des équations de détection de collisions, mais leur *nombre*. Pour cela, nous allons abandonner les sphères englobantes et montrer que la combinaison de l'*arithmétique d'intervalles* et des *boîtes englobantes orientées* permet de détecter des collisions en continu de façon suffisamment efficace pour pouvoir interagir en temps réel avec certains modèles industriels, donnés en exemple. Nous montrerons également que cette combinaison peut être exploitée pour accélérer les méthodes continues existantes qui détectent des collisions entre surfaces paramétrées ou implicites rigides [RKC02b].

Les hiérarchies de volumes englobants perdent en efficacité à mesure que les objets se rapprochent les uns des autres, et que de plus en plus de volumes englobants se recouvrent. Ceci est du au fait que les volumes englobants ne permettent de discriminer les parties des objets qu'en fonction de leur *position*. Nous allons présenter au chapitre 4 une méthode permettant d'exploiter le *mouvement* des objets pour accélérer la détection de collisions. Cette méthode est une généralisation d'une technique de Vanecek [Van94], qui remarque que deux triangles qui s'éloignent l'un de l'autre ne peuvent entrer en contact. La méthode de Vanecek, cependant, effectue les tests de recul au niveau du *triangle*, et dépend donc linéairement du nombre de triangles composant l'objet. Ceci rend la méthode de Vanecek impraticable pour les objets complexes. La généralisation présentée au chapitre 4 effectue les tests de recul au niveau du *volume englobant* et *en temps constant par volume englobant*, pour accélérer significativement la détection de collisions, même pour des objets complexes, en particulier lorsque les objets sont très proches les uns des autres. Cette généralisation est applicable pour n'importe quel type de volume englobant, et pour les méthodes discrètes et continues [RKC02c].

Le chapitre 5 concerne la résolution des problèmes dynamiques. La plupart des méthodes de simulation dynamique par contraintes (*cf* [APS99, Bar94, RK97, SS98a, TPSL95]) formulent les problèmes dynamiques dans *l'espace des contacts*. Ainsi, comme dans l'équation (1.6), les grandeurs physiques explicites sont exprimées aux points de contact. Cependant, ces grandeurs sont redondantes lorsque, par exemple, le nombre de points de contact dans le système dépasse le nombre de degrés de liberté du système.

Aussi, les algorithmes résolvant les problèmes dynamiques formulés dans l'espace des contacts peuvent effectuer des calculs redondants, et perdre en efficacité. Nous montrerons dans ce chapitre que la formulation des problèmes dynamiques dans l'espace des *mouvements*, obtenue à partir du principe des moindres contraintes de Gauss, est équivalente à celle décrite dans l'espace des contacts, mais plus avantageuse sur le plan algorithmique [RKC02a]. Ceci nous amènera à proposer un modèle de friction dans l'espace des mouvements qui, s'il est différent du modèle de Coulomb, semble convenir aux applications d'infographie et de réalité virtuelle.

Les algorithmes de détection de collisions continue et ceux de dynamique ont été implantés et couplés pour former une librairie C++ de simulation dynamique interactive, CONTACT Toolkit. Le chapitre 6 décrit la méthode utilisée pour coupler les algorithmes, ainsi que les principales applications de CONTACT Toolkit, notamment pour la simulation interactive avec retour d'efforts. Plusieurs bases de données industrielles, fournies par Renault et Airbus-EADS, ont ainsi été testées avec succès, sur plusieurs configurations matérielles. Nous verrons dans ce chapitre que, de manière générale, les tests nous ont permis de démontrer le confort apporté par la détection de collisions continue interactive, notamment grâce au réalisme et à la cohérence de la simulation, en particulier en présence de retour d'efforts.

Enfin, le chapitre 7 résumera les principaux points de ce mémoire et présentera quelques directions que nous pensons importantes, pour les développements futurs.

1.6 Conclusion

Dans ce chapitre, nous avons présenté les principes généraux d'un simulateur dynamique interactif, et rappelé les principales méthodes existantes pour détecter des collisions et résoudre les problèmes dynamiques. Nous avons également montré que les méthodes de détection de collisions discrètes, très majoritaires dans les simulateurs, présentent des inconvénients importants.

Nous allons maintenant proposer, au chapitre suivant, la notion de *mouvement intermédiaire arbitraire* pour interpoler les positions successives des objets et détecter des collisions en continu efficacement.

Chapitre 2

Mouvements intermédiaires arbitraires

Afin de détecter efficacement des collisions entre objets polyédriques rigides en continu, nous proposons d'utiliser un mouvement intermédiaire arbitraire pour interpoler les positions successives des objets. Dans une première approche, un premier type de mouvement arbitraire est choisi pour réduire le coût des équations de détection de collisions continues, permettant ainsi de détecter des collisions en continu en temps réel entre objets polyédriques.

2.1 Introduction

Dans le chapitre précédent, nous avons pu constater qu'il existe un réel besoin de méthodes de détection de collisions continues efficaces. Les méthodes discrètes, en effet, peuvent conduire à un manque de réalisme, ainsi qu'à des incohérences et des instabilités dans la simulation.

Nous avons vu, cependant, que le choix d'une méthode de détection de collisions discrète s'explique en quelque sorte naturellement par la difficulté d'utiliser le mouvement réel de l'objet, hormis en certains instant discrets, soit parce qu'il est inconnu (lorsque le calculateur dynamique discrétise les équations de la dynamique, ou bien lorsque l'objet est manipulé par un utilisateur au moyen d'une interface qui n'envoie les actions de l'utilisateur qu'à certains instants), ou parce qu'il conduit à des équations de détection de collisions trop complexes pour pouvoir être résolues efficacement.

Dans ce chapitre et le suivant, nous allons utiliser des *mouvements intermédiaires arbitraires* pour interpoler les positions successives des objets en ces instants discrets et remplacer leur mouvement réel, inconnu ou inexploitable, *afin de résoudre efficacement les équations de détection de collisions.*

En particulier, nous allons montrer, dans une première approche qui s'attachera à réduire les coûts des détections de collisions, qu'un premier type de mouvement

arbitraire, dérivé d'un vissage, permet de réduire les équations de détection de collisions élémentaires, ainsi que les tests de recouvrement continus entre sphères englobantes, à des polynômes de degré inférieur ou égal à trois, qui peuvent être résolus simplement et efficacement grâce aux formules de Cardan.

Ce chapitre est organisé de la manière suivante. Dans la section 2.2, le principe de l'utilisation d'un mouvement intermédiaire arbitraire est introduit et discuté. La section 2.3 va décrire une première classe de mouvements intermédiaires arbitraires, dérivés de vissages, qui vérifient les contraintes exposées dans la section 2.2. Dans la section 2.4, nous montrons comment un certain mouvement de cette classe permet de réduire les équations de détection de collisions élémentaires (*i.e.* entre primitives polyédriques) à des polynômes de degré inférieur ou égal à trois. La section 2.5 montre ensuite que des tests de recouvrements continus entre sphères peuvent également être réduits à des calculs de racines de polynômes de degré trois, en utilisant le même mouvement qu'à la section 2.4. La section 2.6 présente plusieurs optimisations des algorithmes de détection de collisions. Les tests élémentaires et les tests de recouvrement permettent de construire une librairie de détection de collisions complète. Cette librairie est évaluée dans la section 2.7, par des tests qui montrent la validité de l'utilisation d'un mouvement intermédiaire arbitraire en détection de collisions continue. Enfin, la section 2.8 conclut le chapitre.

2.2 Mouvement intermédiaire arbitraire

Nous avons vu dans le premier chapitre que, dans la plupart des cas, les positions des objets ne sont connues qu'à des instants discrets. Dans la plupart des cas en effet, ces positions sont calculées par le simulateur dynamique ou bien imposées par l'interface utilisateur. Dans d'autres cas, le mouvement de l'objet est connu, peut même s'exprimer analytiquement en fonction du temps, mais est trop complexe pour pouvoir résoudre facilement les équations de détection de collisions, et ne peut servir qu'à afficher les objets à des instants discrets. Pour ces raisons, *nous ne pouvons pas exploiter le mouvement réel de l'objet entre ces instants discrets.*

En quelques mots, l'idée centrale de ce chapitre et du chapitre suivant est de remplacer le mouvement de l'objet entre deux instants discrets successifs quelconques, indisponible ou trop complexe, par un mouvement *arbitrairement fixé, qui permette de résoudre efficacement les équations de détection de collisions.*

Bien sûr, ce mouvement intermédiaire arbitraire doit respecter plusieurs contraintes. Notons t_n et t_{n+1} deux instants discrets successifs quelconques, auxquels 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. Le mouvement intermédiaire arbitraire doit être :

– rigide : l'objet doit rester rigide au cours du mouvement[1],

[1]Nous nous autorisons cet abus de langage afin de simplifier le discours. Un terme de substitu-

- interpolant : les positions de l'objet aux deux instants discrets doivent être les mêmes pour le mouvement réel et le mouvement arbitraire,
- continu : comme le mouvement réel de l'objet, le mouvement arbitraire doit 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}.

Bien entendu, une quatrième contrainte est implicite : le mouvement arbitraire doit être *proche du mouvement réel*. Il serait ainsi souhaitable de définir une mesure permettant d'évaluer la différence entre le mouvement réel de l'objet et le mouvement intermédiaire, utilisé pour détecter des collisions entre les instants discrets successifs. Cependant, nous avons vu que ce mouvement réel est rarement disponible. Pour cette raison, nous proposons d'utiliser pour référence le mouvement de l'objet défini à partir de la position et de la vitesse déterminées par le calculateur dynamique. Ainsi, nous avons vu qu'à tous les instants discrets successifs le calculateur dynamique déduit les positions et vitesses des objets à partir de celles de l'instant précédent et éventuellement, pour les schémas d'intégration d'ordre supérieur (comme celui de Runge-Kutta à l'ordre 4), de celles établies à certains instants intermédiaires. Il est possible, comme le fait en quelque sorte le calculateur dynamique, de supposer que les vitesses translationnelle et rotationnelle de l'objet sont constantes durant le pas de temps. Un mouvement arbitraire naturel est alors un mouvement dont les vitesses ne s'éloignent pas trop de ces vitesses de références. Une autre possibilité consisterait à calculer les positions de l'objet en plusieurs instants intermédiaires au moyen des équations de la dynamique et à comparer ces positions aux positions de l'objet aux mêmes instants lorsque le mouvement arbitraire est employé.

Une fois ces conditions (bien naturelles) remplies, cependant, le mouvement intermédiaire peut être arbitrairement choisi *de façon que les équations de détection de collisions puissent être résolues le plus efficacement possible.*

Pour mieux nous représenter la façon dont le mouvement arbitraire est employé, considérons par exemple le cas d'un simulateur dynamique interactif qui calcule la position d'une théière et affiche une nouvelle image tous les trentièmes de seconde. Entre deux images, le mouvement de la théière 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 la théière est modifié, et son mouvement *global* est conservé, comme le montre la figure 2.1. Dans cet exemple, la position de la théière est connue à sept instants discrets $t_0, \ldots, t_6$. Dans la partie gauche ((a), agrandie en (c)), le mouvement réel de la théière entre ces instants est représenté en transparence. Dans la partie droite ((b), agrandie en (d)), les positions connues sont conservées, mais le mouvement réel de l'objet entre deux positions connues successives a été remplacé par un mouvement intermédiaire arbitraire, représenté en transparence. L'allure générale de la trajectoire de la théière est conservée.

Il devrait être clair, maintenant, qu'utiliser un mouvement intermédiaire arbi-

tion, un peu plus général, pourrait être *conservatif*, pour exprimer la nécessité de conserver certaines propriétés de l'objet au cours du mouvement, notamment sa rigidité.

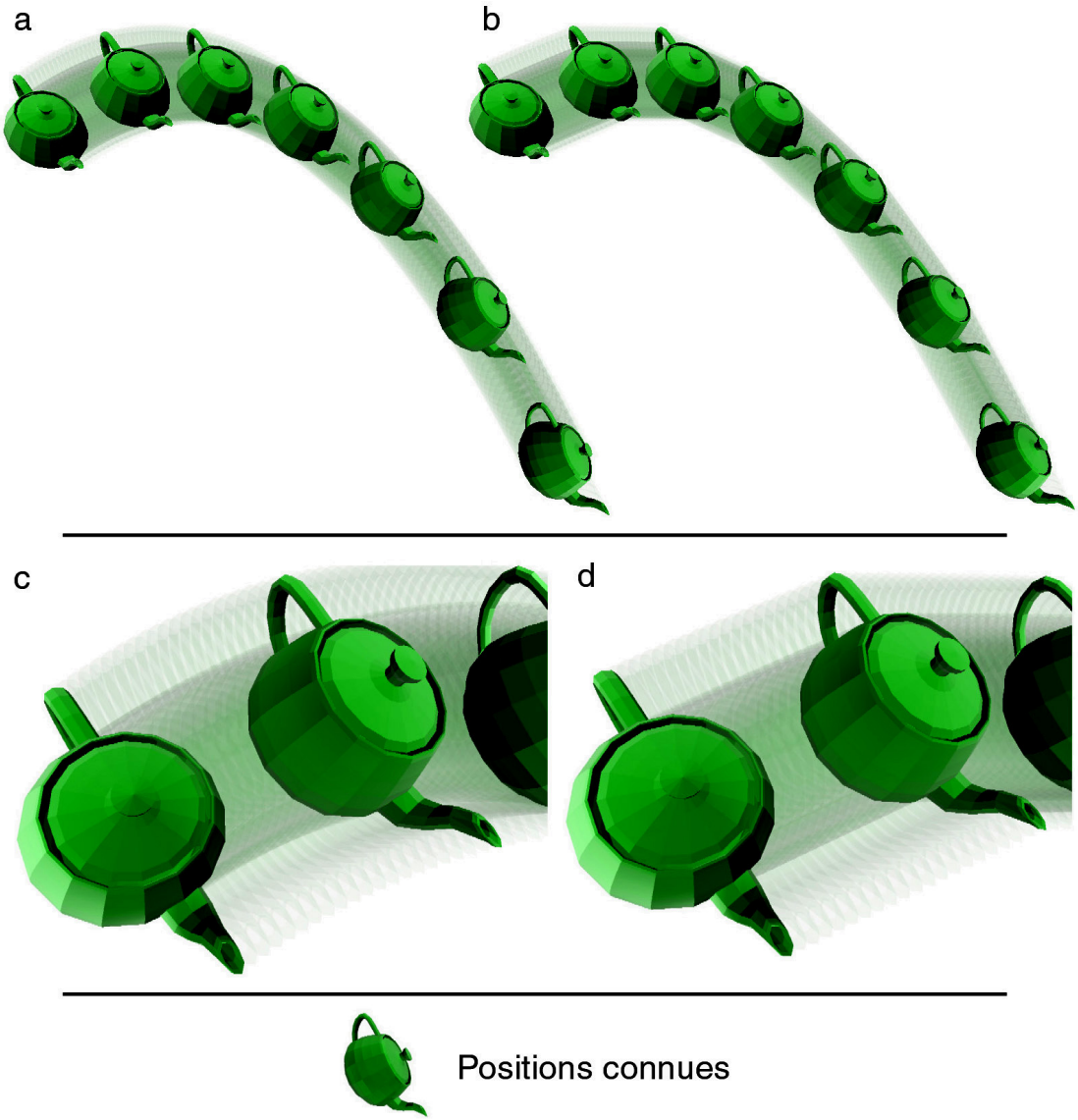


Fig. 2.1 – Utilisation d'un mouvement intermédiaire arbitraire pour interpoler les positions successives connues de la théière. Dans la partie gauche de l'image (a, agrandie en c), sept positions de la théière ont été déterminées par le calculateur dynamique. Entre ces positions, la trajectoire réelle de la théière est représentée. Dans la partie droite de l'image (b, agrandie en d), les sept positions déterminées par le calculateur sont conservées, mais entre ces positions, le mouvement réel de l'objet est remplacé par un mouvement intermédiaire arbitraire. La trajectoire globale de la théière est inchangée.

traire est à peu près équivalent au principe sous-jacent de tout schéma d'intégration des équations différentielles de la dynamique : discrétiser les équations différentielles de la dynamique revient à supposer que les vitesses et accélérations sont constantes en dehors de certains instants discrets.

Nous allons maintenant formaliser les contraintes imposées sur le mouvement intermédiaire arbitraire. Pour cela, notons $\mathbf{P}_R(t)$ la matrice 4×4 décrivant la position *réelle* de l'objet au cours de l'intervalle de temps $[t_n, t_{n+1}]$. Rappelons que cette matrice permet de calculer les coordonnées réelles (homogènes) $\mathbf{x}_R(t)$ d'un point de l'objet dans le repère global à partir de ses coordonnées (homogènes) $\mathbf{x}_o$ dans le repère local de l'objet :

$$\mathbf{x}_R(t) = \mathbf{P}_R(t)\mathbf{x}_o \tag{2.1}$$

Les vecteurs $\mathbf{x}_R(t)$ et $\mathbf{x}_o$ sont des vecteurs homogènes de $\mathbb{R}^4$, dont la dernière coordonnée est le réel 1.

Notons maintenant $\mathbf{P}_A(t)$ la matrice de position de l'objet donnée par le mouvement *arbitraire*, au cours du même intervalle de temps $[t_n, t_{n+1}]$. De la même facon, les coordonnées arbitraires $\mathbf{x}_A(t)$ d'un point de l'objet dans le repère global sont obtenues à partir de ses coordonnées dans le repère local de l'objet :

$$\mathbf{x}_A(t) = \mathbf{P}_A(t)\mathbf{x}_o \tag{2.2}$$

Les trois contraintes se formalisent simplement :

- la contrainte de rigidité impose que la matrice $\mathbf{P}_A(t)$ soit bien une matrice de *position* à chaque instant t compris entre t_n et t_{n+1}. Autrement dit, elle ne doit pas comporter de termes de déformation (changement d'échelle par exemple), et doit être la combinaison d'une matrice de rotation et d'un vecteur de translation, suivant la forme classique d'une matrice homogène de position :

$$\mathbf{P}_A(t) = \begin{pmatrix} \mathbf{R}_A(t) & \mathbf{T}_A(t) \\ \mathbf{0} & 1 \end{pmatrix} \tag{2.3}$$

- la contrainte d'interpolation impose simplement que $\mathbf{P}_A(t_n) = \mathbf{P}_R(t_n)$, ainsi que $\mathbf{P}_A(t_{n+1}) = \mathbf{P}_R(t_{n+1})$,
- la contrainte de continuité impose que la fonction $t \mapsto \mathbf{P}_A(t)$ soit continue sur l'intervalle $[t_n, t_{n+1}]$, autrement dit que les coefficients de $\mathbf{R}_A(t)$ et $\mathbf{T}_A(t)$ soient des fonctions continues du temps sur cet intervalle.

Il est possible de définir rigoureusement la quatrième contrainte que nous avons évoquée précédemment, et qui consiste à choisir un mouvement arbitraire proche du mouvement réel. Il suffirait par exemple de définir l'erreur maximale sur la position des points de l'objet résultant de l'emploi d'un mouvement arbitraire. Nous avons cependant constaté empiriquement que les mouvements arbitraires décrits dans ce mémoire sont toujours suffisamment naturels, pour que l'utilisateur ne soit pas surpris par la position de l'objet qu'il manipule au moment de la collision (comme l'on peut s'y attendre en examinant la figure 2.1).

Pour la même raison, nous n'avons pas ressenti la nécessité de définir des contraintes de continuité et d'interpolation d'ordre supérieur. Pour certaines applications cependant, il est possible qu'il soit souhaitable de contraindre le mouvement intermédiaire à interpoler non seulement les positions de l'objet aux instants discrets

successifs, mais aussi les vitesses en ces mêmes instants, ou même les dérivées d'ordre supérieur.

Notons que, d'après le schéma général d'un simulateur donné au chapitre précédent (figure 1.1), 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.

Sans cela, en effet, des interpénétrations pourraient intervenir, comme le montre la figure 2.2. En (a), une collision a été détectée à l'instant t_c en utilisant le mouvement intermédiaire arbitraire. En (b), le calculateur dynamique a été utilisé pour calculer la position réelle de l'objet à l'instant t_c, ce qui résulte en une interpénétration. Pour la même raison, lorsqu'une collision est détectée, le mouvement arbitraire doit être aussi utilisé pour les objets qui ne sont pas entrés en contact avec un autre objet. Evaluer leurs positions réelles à l'instant t_c grâce au calculateur dynamique pourrait engendrer des interpénétrations, puisque ces positions réelles n'ont pas été employées pour la détection de collisions.

Ainsi, l'utilisation d'un mouvement arbitraire intermédiaire pour détecter des collisions perturbe le cours de la simulation. Il est en effet très peu probable que le mouvement réel des objets et le mouvement intermédiaire arbitraire conduisent à détecter des collisions aux mêmes instants et aux mêmes endroits. Il n'est même pas garanti qu'une collision intervenant entre deux objets lorsque l'un des deux mouvements (réel ou arbitraire) est utilisé intervienne aussi lorsque l'autre mouvement est employé. Cependant, cette méthode permet, comme nous le verrons, de détecter des collisions en continu très efficacement, en conservant les bénéfices d'une méthode continue qui utiliserait le mouvement réel de l'objet (qui, de toutes façons, n'est pas disponible dans la plupart des cas). Avec cette méthode, en effet, les objets sont en permanence dans un état consistant : aucune interpénétration n'est possible et aucune collision ne peut être manquée. Comme nous l'avons remarqué précédemment, nous avons pu constater au cours de nos expériences qu'employer un mouvement arbitraire ne nuit pas au réalisme de la simulation, au moins du point de vue d'un observateur humain.

Par ailleurs, pour certaines applications physiques plus rigoureuses, il devrait suffire de réduire le pas de temps, grâce aux contraintes de continuité et d'interpolation imposées sur le mouvement arbitraire. En effet, les instants entre lesquels le mouvement de l'objet est remplacé ne sont pas nécessairement les instants auxquels les objets sont affichés. Supposons par exemple que l'objet soit très rapide sur l'intervalle de temps $[t_n, t_{n+1}]$. Afin de réduire l'erreur entre le mouvement réel de l'objet et le mouvement arbitraire utilisé pour la détection de collisions, il peut être jugé préférable de subdiviser

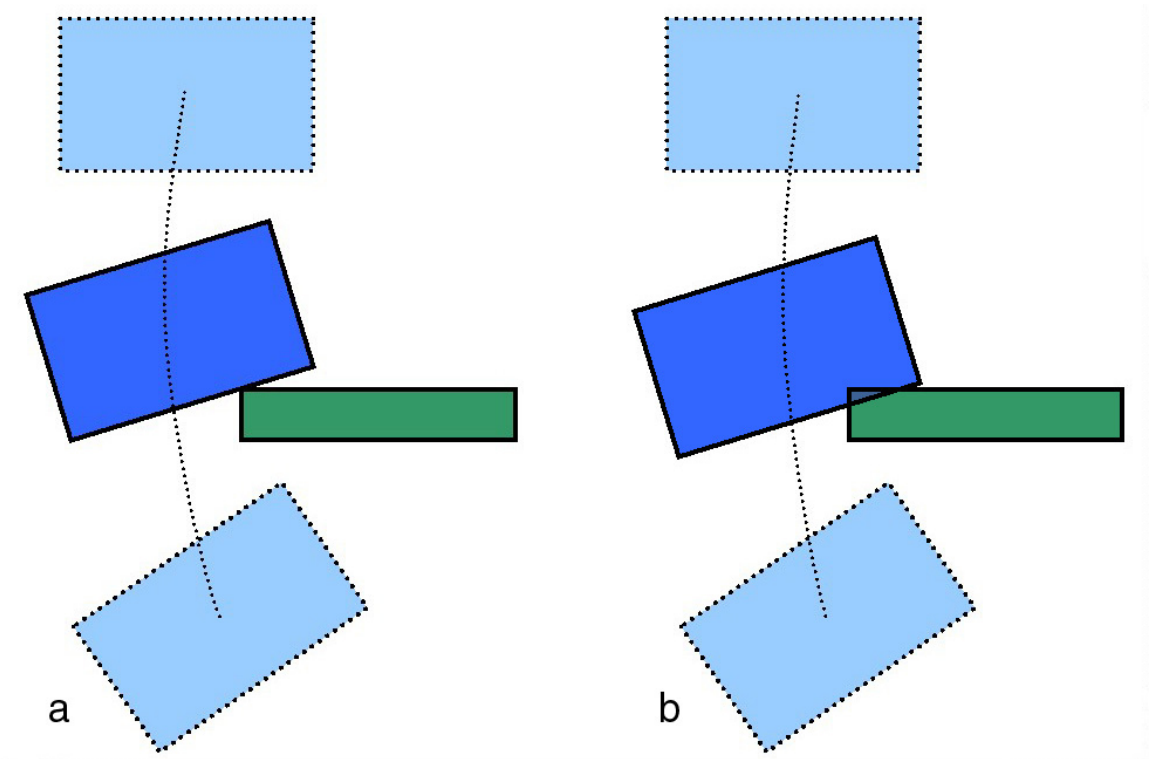


Fig. 2.2 – Afin d'éviter toute interpénétration, il est nécessaire de calculer les positions des objets à l'instant de la collision en employant le mouvement intermédiaire utilisé pour la détection de collisions, et non pas le mouvement réel de l'objet.

l'intervalle de temps en deux sous-intervalles $[t_n, t_i]$ et $[t_i, t_{n+1}]$, où $t_i = \frac{t_n+t_{n+1}}{2}$. Au temps intermédiaire t_i, la position de l'objet est évaluée, par exemple par le simulateur dynamique, puis est utilisée pour remplacer le mouvement réel de l'objet par *deux* mouvements arbitraires successifs, sur $[t_n, t_i]$ puis $[t_i, t_{n+1}]$. Ceci permet de s'assurer que la position intermédiaire au temps t_i, au moins, est celle souhaitée par le simulateur dynamique.

Ainsi, le défaut de précision entraîné par l'utilisation d'un mouvement intermédiaire arbitraire repose au fond sur le même principe d'approximation qui prévaut lors de la discrétisation des équations de la dynamique. Ce défaut, tout relatif, puisque nous avons pu constater le réalisme des simulations effectuées, est largement compensé par les bénéfices que fournit une méthode de détection de collisions continue.

Pour conclure ce paragraphe, il est souhaitable de résumer en quelques mots le principe de l'utilisation d'un mouvement intermédiaire arbitraire : *puisque le mouvement réel de l'objet entre deux instants discrets successifs quelconques n'est pas exploitable pour détecter des collisions en continu, il est remplacé par un mouvement intermédiaire arbitrairement fixé, qui doit respecter trois contraintes : ce mouvement arbitraire doit* interpoler *de façon* rigide *et* continue *les positions de l'objet aux deux instants discrets. Parmi les mouvements arbitraires qui respectent ces contraintes, il faut ensuite en choisir un, proche du mouvement réel de l'objet, qui permette de ré-*

soudre les équations de détection de collisions très efficacement.

2.3 Un mouvement arbitraire dérivé d'un vissage

Dans une première approche, que nous décrivons dans la suite du chapitre, nous allons nous attacher à réduire le coût des équations de détections de collisions, en utilisant un premier type de mouvement intermédiaire arbitraire qui nous permette de résoudre très efficacement les équations de détection de collisions. Nous allons en effet montrer qu'un certain type de mouvement intermédiaire permet de réduire à la fois les équations de détection de collisions élémentaires et les tests de recouvrement entre des sphères englobantes à des polynômes de degré trois.

Pour cela, nous allons dériver le mouvement intermédiaire arbitraire d'un *vissage*, afin d'interpoler les positions successives des objets.

Un vissage $\mathcal{V}(\omega, s, \mathbf{O}, \mathbf{u})$ est la composée commutative d'une rotation et d'une translation de mêmes axes. Les réels ω et s sont respectivement la quantité de rotation et de translation de la transformation, $\mathbf{O}$ est un point de l'axe du vissage, et $\mathbf{u}$ est un vecteur directeur de l'axe. Un exemple de vissage est représenté dans la figure 2.3. Dans cet exemple, le vissage transforme le point $\mathbf{A}$ en le point $\mathbf{A}'$. Selon que la rotation ou la translation est appliquée en premier au point $\mathbf{A}$, le point intermédiaire est respectivement $\mathbf{A}_1$ ou $\mathbf{A}_2$.

L'intérêt des vissages est qu'ils permettent d'interpoler deux positions rigides quelconques d'un objet. En effet, quelles que soient les positions d'un objet aux instants t_n et t_{n+1}, il existe un vissage unique qui transforme la position initiale (*i.e.* à l'instant t_n) en la position finale (*i.e.* à l'instant t_{n+1}) (au choix de $\mathbf{O}$ sur l'axe près, et en imposant à ω d'être positif [Cha1831]). Dans notre implantation, le vissage équivalent au mouvement de l'objet est obtenu directement à partir des vitesses translationnelle et rotationnelle de l'objet (*cf* annexe A).

En théorie, utiliser un vissage pour interpoler deux positions successives pourrait conduire à un mouvement intermédiaire non naturel. Dans la figure 2.4, le mouvement réel de l'objet (à gauche) a été remplacé par le vissage à angle positif équivalent (à droite). Nous verrons cependant dans le chapitre 6 que, dans la pratique, les pas de temps employés dans la simulation sont très petits, et l'utilisateur ne perçoit pas la différence. Pour la même raison, le fait que le vissage n'interpole que les positions des objets et non leurs vitesses, ou même les dérivées d'ordre supérieur du mouvement, n'entrave pas le réalisme de la simulation interactive.

Supposons, sans perte de généralité, que l'intervalle de temps courant est l'intervalle $[0, 1]$. Pour obtenir un mouvement rigide et continu qui interpole les positions initiale et finale, il suffit de faire varier continûment les paramètres ω et s. Ceci peut être réalisé en choisissant deux fonctions $a : \mathbb{R}^2 \times [0, 1] \to \mathbb{R}$ et $b : \mathbb{R}^2 \times [0, 1] \to \mathbb{R}$

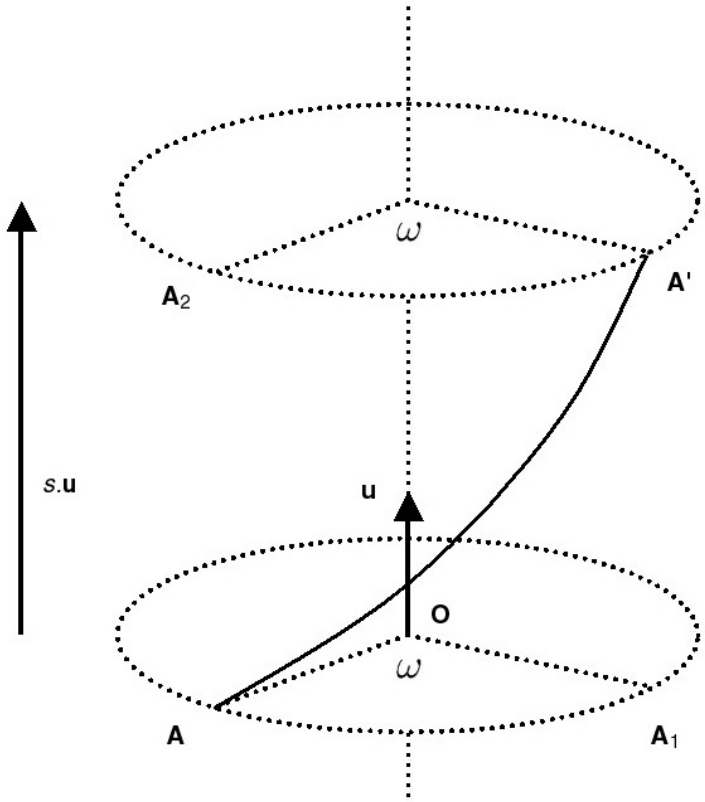


Fig. 2.3 – Un vissage est la composée commutative d'une rotation et d'une translation de mêmes axes.

telles que, pour toute paire (ω, s) dans $\mathbb{R}^2$, les fonctions

$$\begin{aligned} a_{\omega,s} &: \left\{ \begin{array}{c} [0,1] \to \mathbb{R} \\ t \mapsto \omega(t) = a(\omega, s, t) \end{array} \right. \\ b_{\omega,s} &: \left\{ \begin{array}{c} [0,1] \to \mathbb{R} \\ t \mapsto s(t) = b(\omega, s, t) \end{array} \right. \end{aligned} \tag{2.4}$$

soient C^1, monotones, et respectent la condition d'interpolation, c'est-à-dire que $a_{\omega,s}(0) = b_{\omega,s}(0) = 0$, et que $a_{\omega,s}(1) = \omega$ et $b_{\omega,s}(1) = s$.

La classe de mouvements intermédiaires arbitraires est alors de la forme :

$$\mathcal{M} : \left\{ \begin{array}{c} [0,1] \times \mathbb{R}^3 \to \mathbb{R}^3 \\ (t, A) \mapsto A(t) = \mathcal{V}(a_{\omega,s}(t), b_{\omega,s}(t), O, \vec{u})(A_0) \end{array} \right. \tag{2.5}$$

où A_0 est un point de l'objet à l'instant 0 et $A(t)$ le même point au cours du mouvement intermédiaire arbitraire. Il est important de remarquer que les deux fonctions a et b dépendent seulement des paramètres du vissage, et non de la forme de l'objet. Ceci garantit que tous les points de l'objet ont le même mouvement rigide. De plus, grâce aux conditions imposées sur les fonctions $a_{\omega,s}$ et $b_{\omega,s}$, les mouvements arbitraires de la forme (2.5) sont bien rigides, continus et interpolants.

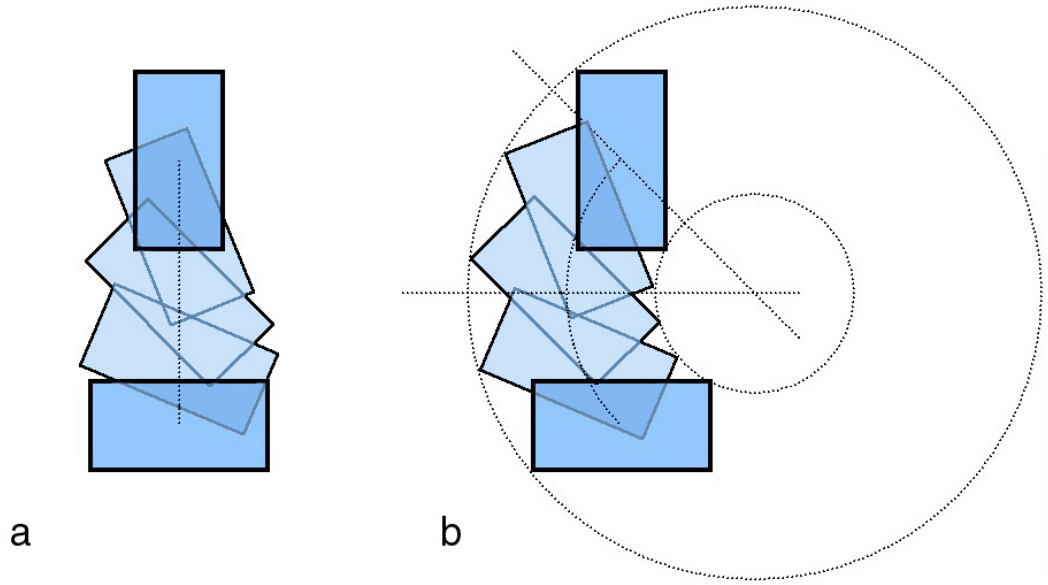


Fig. 2.4 – Utilisation d'un vissage pour remplacer le mouvement le mouvement réel de l'objet. a : le mouvement réel de l'objet est une translation à vitesse constante (de haut en bas) combinée à une rotation à vitesse constante autour du centre de gravité de l'objet. b : le mouvement réel de l'objet a été remplacé par le vissage à angle positif équivalent (et unique). En théorie, l'utilisation d'un vissage comme mouvement intermédiaire arbitraire pourrait conduire à des mouvements non naturels. Dans la pratique, les pas de temps employés dans la simulation sont très faibles et l'utilisateur ne perçoit pas la différence.

Cette description des mouvements intermédiaires est encore un peu formelle. Afin de calculer, dans la pratique, les positions des points de l'objet au cours du temps, il est préférable d'exprimer matriciellement les mouvements (2.5). Définissons tout d'abord un *repère du vissage* comme un repère dont l'axe Oz est l'axe du vissage. Par symétrie axiale, il existe une infinité de tels repères, et il suffit d'en choisir un. Dans l'un de ces repères, le vissage peut être exprimé simplement :

$$\mathbf{V}(t) = \begin{pmatrix} \cos(a_{\omega,s}(t)) & -\sin(a_{\omega,s}(t)) & 0 & 0 \\ \sin(a_{\omega,s}(t)) & \cos(a_{\omega,s}(t)) & 0 & 0 \\ 0 & 0 & 1 & b_{\omega,s}(t) \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.6}$$

pour $t \in [0, 1]$. Dans le repère global, le vissage est alors :

$$\mathbf{S}(t) = \mathbf{P}^{-1}\mathbf{V}(t)\mathbf{P} \tag{2.7}$$

où $\mathbf{V}(t)$ est le vissage d'axe Oz, $\mathbf{P}$ est la matrice de passage du repère global au repère local du vissage, et $\mathbf{P}^{-1}$ est l'inverse de $\mathbf{P}$. Notons que le vissage exprimé dans le repère global est encore une matrice 4×4 homogène.

Grâce à l'expression du vissage dans le repère global (2.7), il est possible d'obtenir les coordonnées des points de l'objet $\mathbf{x}_A(t)$ au cours du mouvement intermédiaire arbitraire :

$$\mathbf{x}_A(t) = \mathbf{P}_A(t)\mathbf{x}_o = \mathbf{P}^{-1}\mathbf{V}(t)\mathbf{P}\mathbf{P}_o\mathbf{x}_o \tag{2.8}$$

où $\mathbf{x}_o$ représente les coordonnées du point dans le repère de l'objet, et $\mathbf{P}_0$ est la matrice de position de l'objet à l'instant 0. La matrice de position de l'objet au cours du mouvement arbitraire est $\mathbf{P}_A(t)$.

Nous allons maintenant montrer comment le choix de deux fonctions a et b particulières va permettre de réduire les équations de détection de collisions élémentaires à des polynômes de degré inférieur ou égal à trois.

2.4 détection de collisions élémentaires

Afin de simplifier le problème de la recherche des fonctions a et b, considérons à partir de maintenant que l'un des deux objets est immobile. Lorsque deux objets sont mobiles, il est possible de considérer leur mouvement relatif. Nous verrons cependant que ceci peut entraîner des problèmes lorsque l'on veut détecter des collisions entre plus de deux objets mobiles, et nous amènera à proposer une deuxième approche dans le chapitre suivant.

Supposons de plus, pour simplifier les calculs, que les coordonnées des sommets sont données dans un repère local du vissage. L'intervalle de temps considéré est, comme précédemment, l'intervalle $[0, 1]$. D'après l'expression du vissage dans le repère local (2.6), les coordonnées homogènes d'un point $\mathbf{a}$ au cours du mouvement sont (dans le repère du vissage) :

$$\mathbf{a}(t) = \begin{pmatrix} a_x(t) \\ a_y(t) \\ a_z(t) \\ 1 \end{pmatrix} = \mathbf{V}(t) \begin{pmatrix} a_x \\ a_y \\ a_z \\ 1 \end{pmatrix} = \begin{pmatrix} a_x \cos(a_{\omega,s}(t)) - a_y \sin(a_{\omega,s}(t)) \\ a_x \sin(a_{\omega,s}(t)) + a_y \cos(a_{\omega,s}(t)) \\ a_z + b_{\omega,s}(t) \\ 1 \end{pmatrix} \tag{2.9}$$

où $a_x(t)$, $a_y(t)$ et $a_z(t)$ sont les coordonnées de $\mathbf{a}(t)$ au cours du mouvement, et a_x, a_y et a_z sont les coordonnées de $\mathbf{a}$ au début du mouvement.

Nous avons vu au chapitre précédent qu'en détection de collisions continue, tous les contacts entre deux objets polyédriques A et B font intervenir au moins l'un des trois types de contact élémentaires non dégénérés :

- une arête de A entre en contact avec une arête de B.
- un point de A entre en contact avec une face de B.
- une face de A entre en contact avec un point de B.

Le troisième type de contact peut être ramené au deuxième en considérant le mouvement opposé des objets, et nous devons donc résoudre deux types d'équations : arête/arête et point/face.

Commençons par le cas arête/arête. En reprenant l'équation (1.3) donnée au chapitre précédent, et en supposant que la deuxième arête est immobile, il faut résoudre l'équation suivante :

$$\mathbf{a}(t)\mathbf{c} \cdot (\mathbf{a}(t)\mathbf{b}(t) \wedge \mathbf{cd}) = 0 \tag{2.10}$$

puisque les points **c** et **d** sont immobiles. En notant $\mathbf{e}(t)$ l'arête mobile $\mathbf{a}(t)\mathbf{b}(t)$ (**e** pour la position initiale **ab**), et **f** l'arête statique **cd**, l'équation (2.10) devient :

$$\mathbf{a}(t)\mathbf{c} \cdot (\mathbf{e}(t) \wedge \mathbf{f}) = 0 \tag{2.11}$$

Il faut maintenant exprimer chacun des termes de cette équation.

D'après (2.9), les coordonnées du vecteur $\mathbf{a}(t)\mathbf{c}$ sont :

$$\mathbf{a}(t)\mathbf{c} = \begin{pmatrix} c_x - a_x \cos(a_{\omega,s}(t)) + a_y \sin(a_{\omega,s}(t)) \\ c_y - a_x \sin(a_{\omega,s}(t)) - a_y \cos(a_{\omega,s}(t)) \\ c_z - a_z - b_{\omega,s}(t) \\ 0 \end{pmatrix} \tag{2.12}$$

De même, les coordonnées du vecteur $\mathbf{e}(t) = \mathbf{a}(t)\mathbf{b}(t)$ sont :

$$\mathbf{e}(t) = \begin{pmatrix} e_x \cos(a_{\omega,s}(t)) - e_y \sin(a_{\omega,s}(t)) \\ e_x \sin(a_{\omega,s}(t)) + e_y \cos(a_{\omega,s}(t)) \\ e_z \\ 0 \end{pmatrix} \tag{2.13}$$

et les coordonnées du produit vectoriel $\mathbf{e}(t) \wedge \mathbf{f}$ sont donc :

$$\mathbf{e}(t) \wedge \mathbf{f} = \begin{pmatrix} f_z(e_x \sin(a_{\omega,s}(t)) + e_y \cos(a_{\omega,s}(t))) - f_y e_z \\ f_x e_z - f_z(e_x \cos(a_{\omega,s}(t)) - e_y \sin(a_{\omega,s}(t))) \\ (f_y e_x - f_x e_x) \cos(a_{\omega,s}(t)) - (f_y e_y + f_x e_x) \sin(a_{\omega,s}(t)) \\ 0 \end{pmatrix} \tag{2.14}$$

En substituant ces dernières expressions dans l'équation initiale (2.11), en réordonnant les termes, et en utilisant l'identité trigonométrique bien connue $\cos^2(x) + sin^2(x) = 1$, on obtient la forme générale de l'équation de détection de collisions entre une arête mobile et une arête statique, lorsque le mouvement de l'arête mobile est dérivé d'un vissage :

$$A_0 \cos(a_{\omega,s}(t)) + A_1 \sin(a_{\omega,s}(t)) + (A_2 \cos(a_{\omega,s}(t)) + A_3 \sin(a_{\omega,s}(t)))b_{\omega,s}(t) + A_4 = 0 \tag{2.15}$$

où A_0, A_1, A_2, A_3 et A_4 sont des coefficients constants qui dépendent des coordonnées des arêtes au début du mouvement :

$$\begin{cases} A_0 = (a_z - c_z)(f_x e_y - f_y e_x) + e_z(a_x f_y - a_y f_x) + f_z(c_x e_y - c_y e_x) \\ A_1 = (a_z - c_z)(f_x e_x + f_y e_y) - e_z(a_x f_x + a_y f_y) + f_z(c_x e_x + c_y e_y) \\ A_2 = f_x e_y - f_y e_x \\ A_3 = f_x e_x + f_y e_y \\ A_4 = e_z(f_x c_y - f_y c_x) + f_z(e_x a_y - e_y a_x) \end{cases} \tag{2.16}$$

Il est possible de constater dès maintenant que, si l'on prend les fonctions a et b les plus simples possibles, qui donnent une vitesse de rotation et de translation constantes :

$$\begin{cases} a(\omega, s, t) = \omega t \\ b(\omega, s, t) = st \end{cases} \tag{2.17}$$

l'équation (2.15) est transcendante, car elle contient à la fois des fonctions trigonométriques et des fonctions polynômiales de t. Cette équation ne peut donc être résolue analytiquement avec de telles fonctions a et b.

Considérons maintenant le cas point/face. Le choix des fonctions a et b sera effectué un peu plus loin. D'après l'équation (1.4) donnée au chapitre précédent, et en considérant que la face **bcd** est immobile, l'équation de détection est :

$$\mathbf{a}(t)\mathbf{b} \cdot \mathbf{n} = 0 \tag{2.18}$$

où $\mathbf{n} = \mathbf{bc} \wedge \mathbf{bd}$ est un vecteur normal à la face (non nécessairement unitaire). Cette équation est légèrement plus simple que la précédente puisque seul le point **a** est mobile.

Les coordonnées du vecteur $\mathbf{a}(t)\mathbf{b}$ sont :

$$\mathbf{a}(t)\mathbf{b} = \begin{pmatrix} b_x - a_x \cos(a_{\omega,s}(t)) + a_y \sin(a_{\omega,s}(t)) \\ b_y - a_x \sin(a_{\omega,s}(t)) - a_y \cos(a_{\omega,s}(t)) \\ b_z - a_z - b_{\omega,s}(t) \\ 0 \end{pmatrix} \tag{2.19}$$

et les coordonnées du vecteur $\mathbf{n}$ sont :

$$\mathbf{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \\ 0 \end{pmatrix} = \begin{pmatrix} (c_y - b_y)(d_z - b_z) - (c_z - b_z)(d_y - b_y) \\ (c_z - b_z)(d_x - b_x) - (c_x - b_x)(d_z - b_z) \\ (c_x - b_x)(d_y - b_y) - (c_y - b_y)(d_x - b_x) \\ 0 \end{pmatrix} \tag{2.20}$$

En substituant ces expressions dans l'équation (2.18), et en réordonnant les termes, on obtient la forme générale de l'équation de détection de collisions entre un point mobile et une face immobile, lorsque le mouvement du point est dérivé d'un vissage :

$$B_0 \cos(a_{\omega,s}(t)) + B_1 \sin(a_{\omega,s}(t)) + B_2 b_{\omega,s}(t) + B_3 = 0 \tag{2.21}$$

où B_0, B_1, B_2 et B_3 sont des coefficients constants qui dépendent des coordonnées du point et des sommets de la face :

$$\begin{cases} B_0 = -a_x n_x - a_y n_y \\ B_1 = a_y n_x - a_x n_y \\ B_2 = -n_z \\ B_3 = b_x n_x + b_y n_y + (b_z - a_z) n_z \end{cases} \tag{2.22}$$

Remarquons qu'ici aussi l'équation de détection est transcendante lorsque les vitesses de rotation et de translation sont constantes, et ne peut alors être résolue analytiquement.

Il est temps de décrire des fonctions a et b qui permettent de réduire les équations de détection de collisions (2.15) et (2.21). Essentiellement, il suffit de transformer les fonctions trigonométriques en des fonctions rationnelles d'un autre paramètre, en faisant un changement de variables classique.

Ainsi, en posant :

$$\begin{cases} a(\omega, s, t) = \omega t \\ b(\omega, s, t) = s f(t) \end{cases} \tag{2.23}$$

où f à la forme suivante :

$$f(t) = \begin{cases} t & \text{si } \omega = 0 \\ \frac{\tan(\omega t/2)}{\tan(\omega/2)} & \text{sinon} \end{cases} \tag{2.24}$$

les équations (2.15) et (2.21) sont polynomiales en la variable t, lorsque le mouvement est une translation pure (ce qui est facile à vérifier), ou bien en la variable $\tau = \tan(\omega t/2)$, lorsque le mouvement comprend une quantité de rotation non nulle.

En effet, lorsque ω est non nul, les fonctions trigonométriques peuvent être paramétrées de façon classique :

$$\begin{cases} \tau = \tan(\omega t/2) \\ \cos(\omega t) = \dfrac{1-\tau^2}{1+\tau^2} \\ \sin(\omega t) = \dfrac{2\tau}{1+\tau^2} \end{cases} \tag{2.25}$$

L'équation générale de détection de collisions entre une arête statique et une arête mobile (2.15) devient alors :

$$A_0 \frac{1-\tau^2}{1+\tau^2} + A_1 \frac{2\tau}{1+\tau^2} + \left(A_2 \frac{1-\tau^2}{1+\tau^2} + A_3 \frac{2\tau}{1+\tau^2} \right) \frac{s}{\tan(\omega/2)} \tau + A_4 = 0 \tag{2.26}$$

En réduisant au même dénominateur, et en réordonnant les termes, on obtient une équation polynomiale en τ de degré inférieur ou égal à trois :

$$-\frac{sA_2}{\tan(\omega/2)}\tau^3 + \left(\frac{2sA_3}{\tan(\omega/2)} + A_4 - A_0 \right) \tau^2 + \left(2A_1 + \frac{sA_2}{\tan(\omega/2)} \right) \tau + A_0 + A_4 = 0 \tag{2.27}$$

De la même façon, il est possible de réduire l'équation de détection de collisions entre un point et une face (2.21) à une équation polynomiale en τ, grâce aux paramétrisations (2.25). On obtient alors :

$$B_0 \frac{1-\tau^2}{1+\tau^2} + B_1 \frac{2\tau}{1+\tau^2} + B_2 \frac{s}{\tan(\omega/2)} \tau + B_3 = 0 \tag{2.28}$$

En réduisant au même dénominateur et en ordonnant les termes, on obtient bien une équation polynomiale de degré inférieur ou égal à trois :

$$\frac{sB_2}{\tan(\omega/2)}\tau^3 + (B_3 - B_0)\tau^2 + \left(2B_1 + \frac{sB_2}{\tan(\omega/2)}\right)\tau + B_3 + B_0 = 0 \tag{2.29}$$

Ainsi, les fonctions a et b particulières données dans les équations (2.23) et (2.24) permettent de réduire les équations de détection de collisions élémentaires à des polynômes de degré inférieur ou égal à trois. Les racines de ces polynômes peuvent être calculées très efficacement par les formules de Cardan, par exemple.

2.5 Tests de recouvrement entre hiérarchies de sphères

Bien sûr, il n'est pas suffisant de résoudre efficacement les équations de détection de collisions élémentaires, puisque tester toutes les paires possibles serait extrêmement inefficace. Nous avons vu au chapitre précédent qu'une méthode très répandue en détection de collisions pour éviter de nombreux tests élémentaires non pertinents consiste à utiliser des hiérarchies de volumes englobants.

Afin de conserver l'esprit du début de ce chapitre, nous souhaitons maintenant choisir un type de volume englobant qui nous permette d'effectuer des tests de recouvrement très efficaces. Bien entendu, les hiérarchies de volumes englobants doivent avoir le même mouvement que l'objet. Autrement dit, nous devons utiliser les mêmes fonctions a et b que celles données dans la section précédente.

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

Rappelons que, dans le cas discret, les sphères ont par exemple été utilisées par Quinlan [Qui94] et Hubbard [Hub95].

Deux sphères A et B se recouvrent si et seulement si la distance entre leurs centres $\mathbf{a}(t)$ et $\mathbf{b}(t)$ devient inférieure ou égale à la somme de leurs rayons r_A and r_B, ce qui s'écrit formellement :

$$||\mathbf{b}(t) - \mathbf{a}(t)|| \leqslant r_A + r_B, \tag{2.30}$$

ou bien, de façon équivalente :

$$||\mathbf{b}(t) - \mathbf{a}(t)||^2 \leqslant (r_A + r_B)^2 \tag{2.31}$$

plus facile à manipuler, puisque $||\mathbf{b}(t) - \mathbf{a}(t)||^2 = \mathbf{a}(t).\mathbf{a}(t) + \mathbf{b}(t).\mathbf{b}(t) - 2\mathbf{a}(t).\mathbf{b}(t)$.

Le cas où le mouvement est une translation pure est extrêmement simple, et se réduit à une équation du second degré. Lorsque ω est non nul, on procède comme précédemment. En considérant que seule la première sphère est mobile, que les coordonnées des centres sont données dans un repère local du vissage, et que les mêmes fonctions a et b sont utilisées, l'inéquation (2.31) est réduite à une inéquation polynomiale de degré inférieur ou égal à quatre :

$$C_4\tau^4 + C_3\tau^3 + C_2\tau^2 + C_1\tau + C_0 \leqslant 0 \tag{2.32}$$

où les coefficients C_0, C_1, C_2, C_3 et C_4 sont calculés en fonction des positions des points au début du mouvement et des paramètres du vissage :

$$\begin{cases} C_4 = \left(\dfrac{s}{\tan(\omega/2)}\right)^2 \\ C_3 = 2\dfrac{s(a_z - b_z)}{\tan(\omega/2)} \\ C_2 = ||\mathbf{ab}||^2 - (r_A + r_B)^2 + 4(b_x a_x + b_y a_y) + \left(\dfrac{s}{\tan(\omega/2)}\right)^2 \\ C_1 = 2\left(\dfrac{s(a_z - b_z)}{\tan(\omega/2)} + 2(b_x a_y - b_y a_x)\right) \\ C_0 = ||\mathbf{ab}||^2 - (r_A + r_B)^2 \end{cases} \tag{2.33}$$

Il est bien sûr possible de résoudre exactement les équations polynomiales de degré quatre, mais, dans notre cas, ce n'est pas nécessaire. En effet, nous avons seulement besoin de détecter les *recouvrements* entre les sphères, et non pas de calculer les instants de premier contact entre elles. Pour cela, il suffit de dériver le polynôme de l'inéquation (2.32) et de calculer les racines du polynôme de troisième degré obtenu. Ces racines permettent d'évaluer les minima et maxima locaux de la fonction (2.32) sur l'intervalle de temps $[0, 1]$, et donc de savoir si les sphères se recouvrent ou non durant cet intervalle.

Il est donc possible, en utilisant les deux fonctions particulières a et b décrites dans ce chapitre, de réduire à la fois les tests de détection de collisions élémentaires et les tests de recouvrement entre sphères englobantes à des polynômes de degré trois, qui peuvent être résolus très efficacement. Nous disposons donc maintenant des outils de base pour détecter des collisions en continu entre objets polyédriques.

Avant de présenter quelques optimisations possibles de la détection de collisions, attardons nous un instant sur les *hiérarchies* de sphères.

Il existe plusieurs méthodes pour construire des hiérarchies de sphères englobantes [Qui94, Hub95, RKK97, BS02]. Nous avons choisi une méthode semblable à celle proposée par Quinlan [Qui94] et détaillée dans Ruspini *et al.* [RKK97]. Dans cette approche *bottom-up*, un objet est recouvert dans un premier temps de petites sphères, qui seront les feuilles de la future hiérarchie. Toutes ces petites sphères ont le

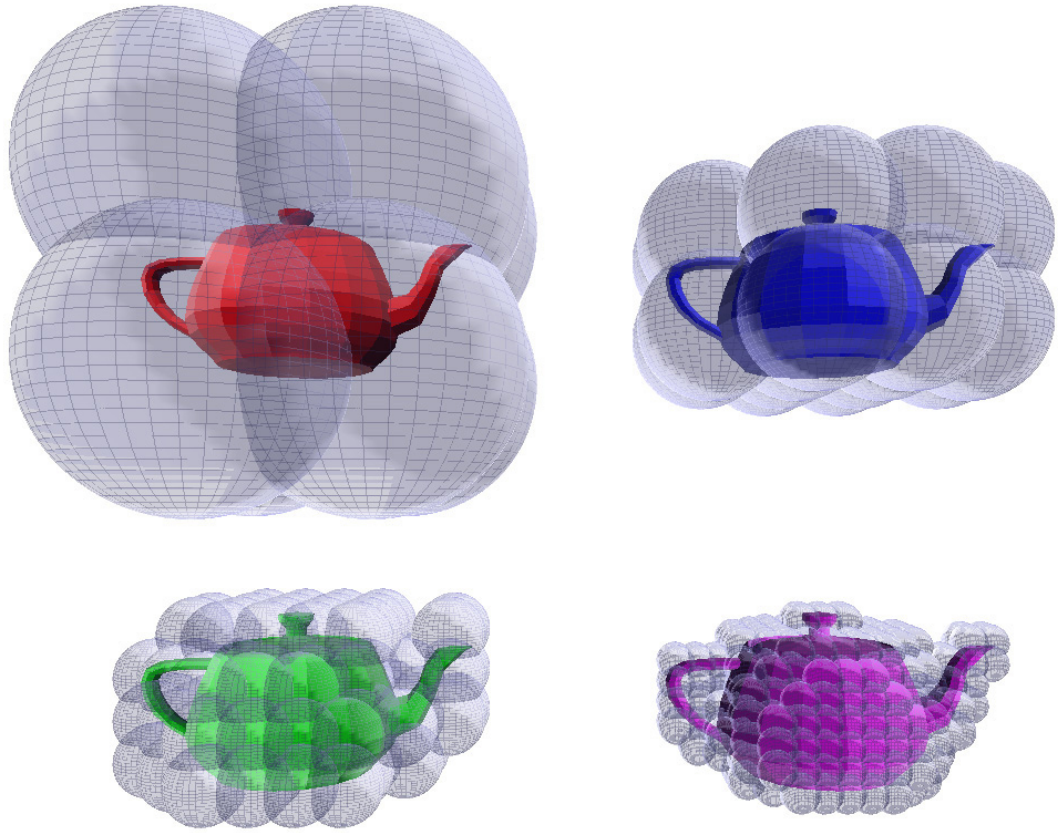


Fig. 2.5 – La disposition des sphères feuilles varie avec leur rayon r_f.

même rayon r_f, définie par l'utilisateur, et qui est *a priori* indépendante de la taille des primitives polyédriques (arêtes ou faces). Ainsi, une sphère feuille peut contenir plusieurs sommets, plusieurs (morceaux d') arêtes, et plusieurs (morceaux de) faces. Inversement, une arête ou une face peut être recouverte par un nombre arbitraire de sphères feuilles. Une règle traditionnelle, cependant, veut que les sphères feuilles ne contiennent qu'un faible nombre de (morceaux de) primitives. Une fois que les sphères feuilles sont créées, une méthode de type *divide-and-conquer* crée les noeuds internes de la hiérarchie (qui sont des sphères, aussi), pour construire un arbre complet [RKK97]. La figure 2.5 montre comment la disposition des sphères feuilles de la hiérarchie varie lorsque l'utilisateur modifie leur rayon r_f.

2.6 Optimisations

Les algorithmes de détection de collisions que nous venons de décrire ont été implantés en C++ pour former une librairie de détection de collisions. Nous présentons maintenant plusieurs optimisations permettant d'accélérer la détection de collisions.

2.6.1 Mise en cache des coordonnées des sommets

A chaque fois qu'une détection de collisions élémentaire est effectuée, les coordonnées des points caractérisant les primitives (c'est-à-dire les points, les extrémités des arêtes ou les sommets des faces) doivent être exprimées dans le repère du vissage correspondant au mouvement de l'objet. Rappelons en effet que, même lorsque les deux objets sont mobiles en même temps, on considère leur mouvement relatif. C'est alors ce mouvement relatif qui est remplacé par un mouvement intermédiaire dérivé d'un vissage.

Malheureusement, ces points peuvent appartenir à plusieurs primitives, et certaines primitives peuvent être situées dans plusieurs sphères feuilles englobantes. De plus, une sphère appartenant à l'objet A peut recouvrir plusieurs sphères appartenant à l'objet B. Afin de ne pas avoir à recalculer les coordonnées des points dans le repère du vissage à chaque fois qu'elles sont nécessaires, ces coordonnées sont stockées dans un tableau.

Comme les fonctions de détection de collisions ne traitent qu'une paire d'objets à la fois, il est possible de stocker ces coordonnées temporaires dans deux tableaux statiques (un pour chaque objet), de taille bornée `nmax` et supérieure au nombre maximal de points dans un objet :

```
cVertex cacheA[nmax]; // coordonnées des points de A dans le repère du vissage
cVertex cacheB[nmax]; // coordonnées des points de B dans le repère du vissage
```

Afin de savoir si, au cours du traitement d'une paire d'objets, un sommet à déjà été transféré dans le repère du vissage, nous employons une méthode classique : un entier `cdstamp` est incrémenté à chaque fois qu'une nouvelle paire d'objets est traitée, et deux tableaux d'entiers (un pour chaque objet), également de tailles `nmax`, sont utilisés :

```
unsigned int stampA[nmax]; // pour l'objet A
unsigned int stampB[nmax]; // pour l'objet B
```

Dès qu'un point d'indice i, appartenant à l'objet A, doit être transféré dans le repère du vissage, on commence par examiner la valeur de `stampA[i]`. Si cette valeur est égale à `cdstamp`, alors le point à déjà été transféré et ses coordonnées sont lues dans `cacheA[i]`. Dans le cas contraire, ses coordonnées dans le repère du vissage sont calculées et stockées dans `cacheA[i]`, et l'entier `stampA[i]` devient égal à `cdstamp`. Cette méthode permet de ne pas avoir à réinitialiser entièrement les tableaux `stampA` et `stampB` à chaque fois qu'une nouvelle paire d'objets est traitée. Nous utiliserons dans le chapitre 4 une méthode semblable, et employant de nouveau l'entier `cdstamp`.

2.6.2 Délimitation de la trajectoire de la sphère

Dans le cas général d'un test de recouvrement entre sphères (lorsque ω et s sont non nuls), où le degré du polynôme à résoudre est trois, il est possible de détecter, tout d'abord, un recouvrement entre un volume englobant la trajectoire de la sphère mobile et la deuxième sphère. Ce volume englobant est le cylindre d'axe Oz dans le repère du vissage. La plupart du temps, ceci permet d'éviter la résolution de l'équation générale.

Enfin, dans les cas particuliers (rotation pure ou translation pure), il est possible d'effectuer les tests de recouvrement sans avoir à effectuer de divisions entre des flottants.

De plus, avant de calculer les racines du polynôme de degré trois, il est aussi très facile de déterminer si les sphères se recouvrent à l'instant initial ou final, en évaluant le polynôme de degré quatre en $\tau = 0$ (position initiale) et $\tau = \tan(\omega/2)$ (position finale). Comme l'on cherche seulement à savoir si les sphères se recouvrent dans l'intervalle de temps $[0, 1]$, il n'est pas nécessaire de poursuivre les calculs lorsque les sphères se recouvrent aux positions initiale ou finale.

2.6.3 Passage de l'instant courant de collision

En détection de collisions continue, seul le *premier* instant de collision dans $[0, 1]$ est intéressant (ainsi que les primitives qui entrent en contact à cet instant). Ainsi, durant le parcours récursif des hiérarchies englobantes, il est nécessaire de maintenir un *instant de première collision* courant t_c. Au début de la détection de collisions entre les deux objets, cet instant courant a une valeur signifiant qu'aucune collision n'a pour l'instant été trouvée (par exemple -1, puisque qu'un instant de collision valide se trouve nécessairement dans l'intervalle $[0, 1]$). Ensuite, à chaque fois qu'une collision est détectée entre deux primitives (*i.e.* entre deux arêtes, ou entre un point et une face), t_c est mis à jour si la nouvelle collision intervient dans l'intervalle $[0, t_c]$ (si t_c vaut -1, alors t_c devient égal au nouvel instant de collision).

Il est possible d'accélérer la détection de collisions en passant la valeur courante de t_c aux fonctions de tests élémentaires et aux fonctions de tests de recouvrement, afin de ne détecter des collisions ou des recouvrements que dans l'intervalle $[0, t_c]$. Ceci est particulièrement utile dans le cas des tests de recouvrements, puisque de nombreux tests élémentaires peuvent être évités.

Ainsi, dans le cas, par exemple, d'un objet très rapide qui tente de passer complètement au travers d'un obstacle fin, il faudrait en effet, en théorie, tester toutes les sphères de la hiérarchie de l'objet mobile (et probablement plusieurs fois). Avec cette méthode, une fois qu'une première collision a été détectée entre deux primitives à l'instant t_c, les fonctions de détection de collisions ne testent plus que l'intervalle de temps $[0, t_c]$. Sur cet intervalle de temps, l'objet à un mouvement réduit, et une grande partie des sphères qui l'englobent n'entrent plus en contact avec des sphères englobantes de

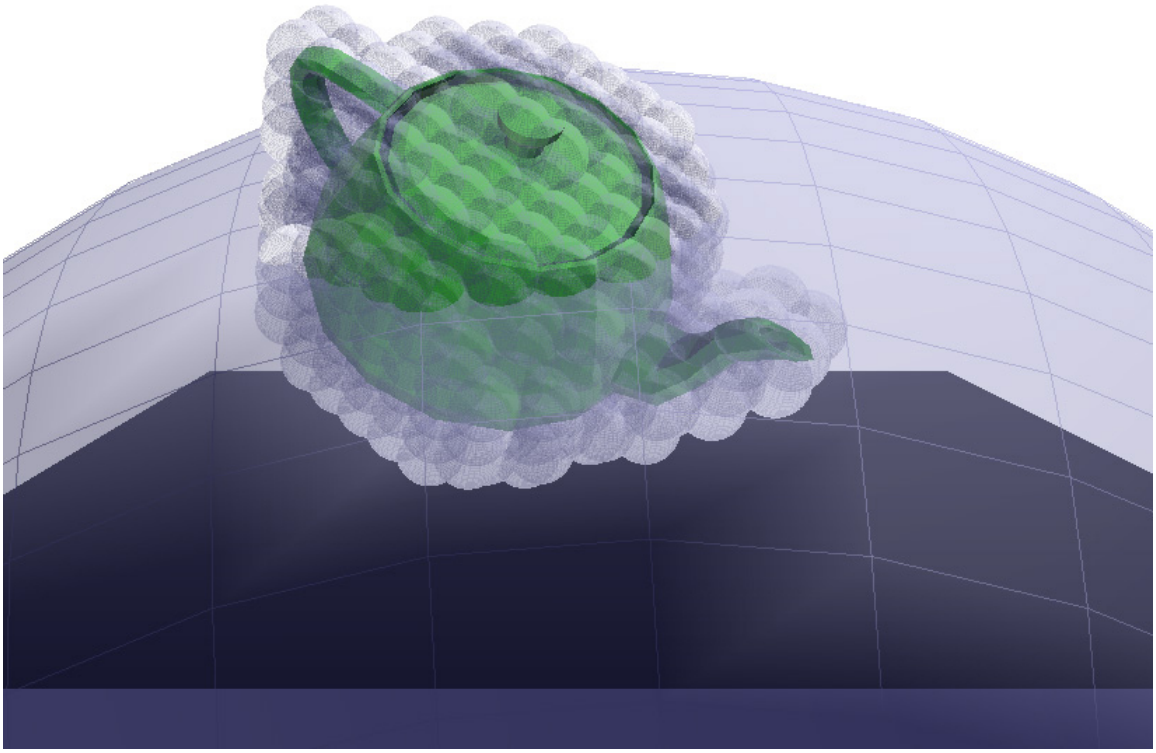


Fig. 2.6 – La théière est à moitié entrée dans la grande sphère feuille du cube (situé sous la théière), ce qui conduit à de nombreux tests élémentaires. Pour éviter cette situation, toutes les sphères feuilles de la scène doivent avoir a peu près la même taille.

l'obstacle. De nombreux autres tests, élémentaires ou de recouvrement, peuvent alors être évités.

2.6.4 La méthode des étiquettes

Finalement, une optimisation s'est révélée particulièrement efficace dans certaines situations. Elle est fondée sur la cohérence spatiale des hiérarchies de volumes englobants.

Dans un environnement virtuel composé d'objet mobiles et immobiles, en effet, une règle simple gouverne le choix des sphères feuilles pour chaque objet : elles doivent être à peu près de même taille, pour tous les objets. Il n'est pas suffisant qu'elles ne contiennent qu'un faible nombre de primitives, puisqu'une sphère feuille d'un objet A peut contenir de nombreuses sphères feuilles d'un objet B, comme le montre la figure 2.6.

Dans cet exemple, A est un simple cube, dont la hiérarchie englobante ne contient qu'une sphère feuille, qui englobe la totalité du cube, et B est une théière, dont la hiérarchie englobante est de profondeur 5 (seules les sphères feuilles sont dessinées). Puisque la théière est à moitié entrée dans la sphère feuille du cube, une grande partie de ses sphères feuilles doit être testée, faisant ainsi perdre l'intérêt d'utiliser un arbre de profondeur 5 pour englober la théière et accélérer la détection de collisions.

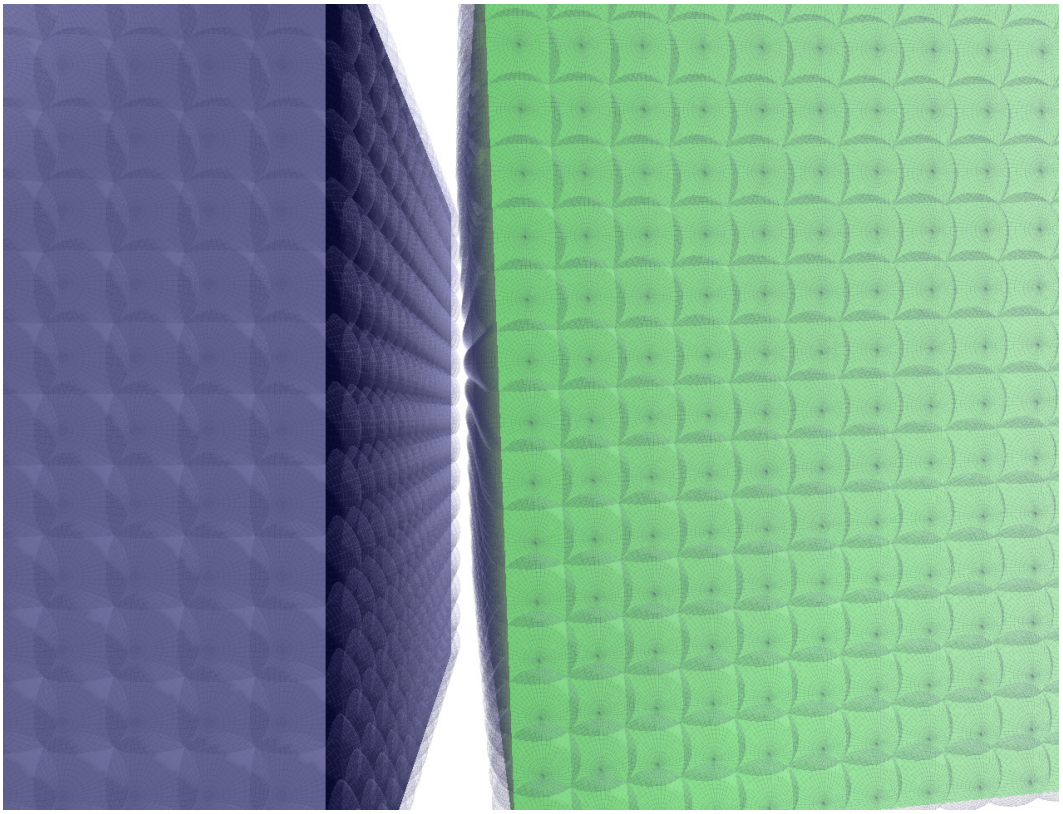


Fig. 2.7 – Une situation problématique sans les étiquettes. De nombreuses paires de sphères feuilles en contact conduisent à tester plusieurs fois la même paire de faces.

Cependant, cette règle peut conduire à ce que de nombreuses sphères feuilles recouvrent une seule face. Si deux faces de ce type, recouvertes par un grand nombre de sphères feuilles, et appartenant chacune à un objet, se rapprochent l'une de l'autre comme dans la figure 2.7, alors l'algorithme de parcours des hiérarchies englobantes va tester la paire de faces autant de fois que de paires de sphères feuilles se recouvrant.

Un moyen d'éviter ceci est d'ajouter une *étiquette* aux noeuds de la hiérarchies. Deux noeuds feuilles qui contiennent le même ensemble de primitives ont la même étiquette (un entier par exemple). Pour les noeuds internes, une règle simple est utilisée pour étiquetter récursivement la hiérarchie. Si tous les fils d'un noeud ont la même étiquette, alors la même étiquette est donnée à ce noeud. Dans le cas contraire, une nouvelle étiquette, différente de toutes celles utilisées jusqu'a présent pour étiquetter l'arbre, est utilisée (il suffit de transmettre à la fonction récursive le numéro de la dernière étiquette employée).

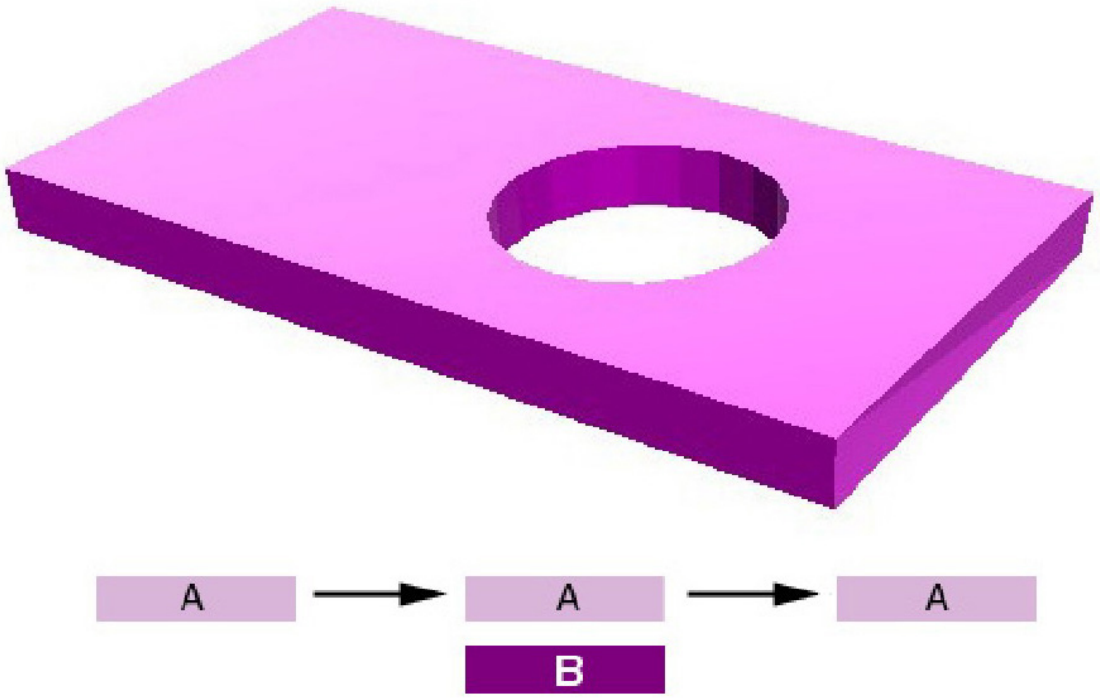


Fig. 2.8 – Deux objets A et B identiques à celui représenté ici ont été utilisés pour tester la méthode des étiquettes. Dans le mouvement employé pour le test, l'objet A se rapproche de l'objet B puis s'en éloigne.

Lorsque deux noeuds doivent subir un test de recouvrement, l'algorithme de parcours des hiérarchies détermine si la paire formée par leurs étiquettes a déjà été testée. Si c'est le cas, la fonction se termine. Il n'est pas nécessaire de parcourir la descendance des noeuds, même s'ils se recouvrent. Sinon (si leur paire n'a pas déjà été testée), le test de recouvrement est effectué, et le parcours des hiérarchies se poursuit de façon habituelle.

Lorsque deux noeud feuilles ont été complètement testés (*i.e.* lorsque tous les tests élémentaires faisant intervenir les primitives qu'ils contiennent ont été effectués), la paire formée par leurs étiquettes est marquée, pour signifier qu'elle a été testée.

Une table de hachage est utilisée pour stocker (et retrouver) les paires d'étiquettes testées. Pour compléter cette technique, trois tables de hachage supplémentaires sont employées pour stocker les paires de primitives testées (une pour les tests arête/arête, une pour les tests point/face, et une pour les tests face/point).

Cette optimisation a permis de réduire significativement le temps total de calcul dans les cas semblables à celui de la figure 2.7. Un test simple a consisté à utiliser deux objets indentiques à celui de la figure 2.8. L'objet B est immobile, tandis que l'objet A, mobile, suit un chemin enregistré. Au début du mouvement, l'objet A est éloigné de l'objet B. Il s'en rapproche au milieu du mouvement pour s'en éloigner de nouveau,

comme décrit dans la figure 2.8.

Le tableau 2.1 donne le nombre total de tests sphère/sphère (S/S), arête/arête (A/A), point/face (P/F) et face/point (F/P) sur l'ensemble de la trajectoire enregistrée, selon les tables de hachage utilisées. Il apparaît que, dans cet exemple, la table de hachage S/S seule permet d'éliminer 17% des tests de détection de collisions, et que les quatre tables de hachage employées ensemble permettent d'éliminer plus de 20% des tests.

	Pas de tables	S/S	A/A-P/F-F/P	S/S-A/A-P/F-F/P
Tests S/S	1.213.596	1.017.112	1.213.596	1.017.112
Tests A/A	87.284	61.916	20.717	20.717
Tests P/F	3.963	3.155	2.367	2.367
Tests F/P	3.858	3.098	2.317	2.317
Total	1.308.701	1.085.281	1.238.997	1.042.513

Tab. 2.1 – La méthode des étiquettes permet de réduire le nombre de tests de collisions et de tests de recouvrements.

Notons qu'un autre avantage de la méthode des étiquettes est qu'elle permet de factoriser la hiérarchie englobante de l'objet et ainsi de réduire la quantité totale de mémoire nécessaire pour stocker la hiérarchie. Par exemple, lorsque deux noeuds feuilles ont la même étiquette, il est possible de stocker une seule fois la liste des primitives contenues dans la feuille. L'étiquette de la feuille sert alors de pointeur vers la liste des primitives.

Remarquons enfin que cette méthode exploite seulement le fait que la structure d'accélération, la hiérarchie englobante, est un arbre, et ne dépend pas de la nature des volumes englobants.

2.7 Discussion

2.7.1 Evaluation des algorithmes

Afin d'évaluer d'une part, l'utilisation d'un mouvement intermédiaire arbitre en détection de collisions continue et, d'autre part, le mouvement arbitraire spécifique que nous venons de décrire, nous avons procédé à plusieurs tests de la librairie de détection de collisions. Essentiellement, nous avons pu constater les avantages d'une méthode de détection de collisions continue dans le cadre de la simulation dynamique en temps réel, et dans celui de la manipulation d'objets.

2.7.2 Simulation dynamique temps réel

Une première série de tests a été conçue pour valider l'utilisation d'un mouvement intermédiaire arbitraire en détection de collisions, en particulier lorsque le réalisme de la simulation doit être respecté. Pour cela, nous avons dans cette première série de tests implanté les lois simples du rebond d'objets rigides. Ces lois sont bien connues [Mac60], et très faciles à implanter lorsque l'instant de collision et la position du contact sont connus, comme c'est le cas lorsque une méthode de détection de collisions continue est utilisée. Nous avons donc pu faire rebondir des objets (un à la fois) sur un sol virtuel.

Les algorithmes de détection de collision se sont révélés assez efficaces pour faire rebondir des objets comportant plusieurs milliers de faces en temps réel, c'est-à-dire avec un taux de rafraîchissement supérieur au taux de rafraîchissement de la carte graphique. Le pas de temps était fixé à un trentième de seconde, et permettait d'obtenir un mouvement physiquement réaliste. De plus, le fait qu'aucune méthode de *backtracking* n'ait due être employée pour replacer les objets dans une position réaliste à chaque rebond (et localiser le contact) aidait à maintenir un taux de rafraîchissement élevé.

Notons par ailleurs que les simulations se sont révélées être *extrêmement robustes* (pas de comportement incohérent, ni d'instabilités, et ce même avec des vitesses initiales aléatoires), ce qui est de première importance dans une simulation dynamique interactive. Nous avons pu constater que certains produits commerciaux, même pour de telles simulations, doivent être configurés attentivement pour produire des simulations réalistes. Ces produits utilisent en effet une méthode détection de collision discrète et leur module de réponse à la collision ne semble pas capable de gérer les arêtes non partagées, ou les surfaces pures qui ne sont pas des volumes, comme la théière de la figure 2.9.

Les tests de nos algorithmes effectués avec cette théière, qui pose des problèmes dans les produits commerciaux évoqués, nous ont ainsi permis de constater la robustesse de la simulation utilisant les algorithmes de détection de collisions continue.

2.7.3 Manipulation d'objets

Une deuxième série de tests a été conçue pour montrer le confort et le réalisme obtenus en utilisant une méthode de détection de collisions continue interactive. Pour cela, une application simple à été programmée, permettant de manipuler un objet et de le faire entrer en contact avec un objet immobile. La manipulation de l'objet était effectuée au clavier sur un PC et grâce à un périphérique à six degrés de liberté (une SpaceBall^{TM}) sur une SGI. Puisque, dans cette série de tests, le but était d'évaluer l'efficacité de la détection de collisions continue avec le mouvement arbitraire spécifique présenté dans les sections précédentes, et non pas la *réponse à la collision*, l'objet mobile était stoppé à chaque fois qu'il entrait en contact avec l'objet statique. L'utilisateur avait alors la possibilité de se déplacer dans la scène, afin de vérifier l'exactitude de la détection de collisions, et de constater que les objets étaient bien *en contact*, et ne

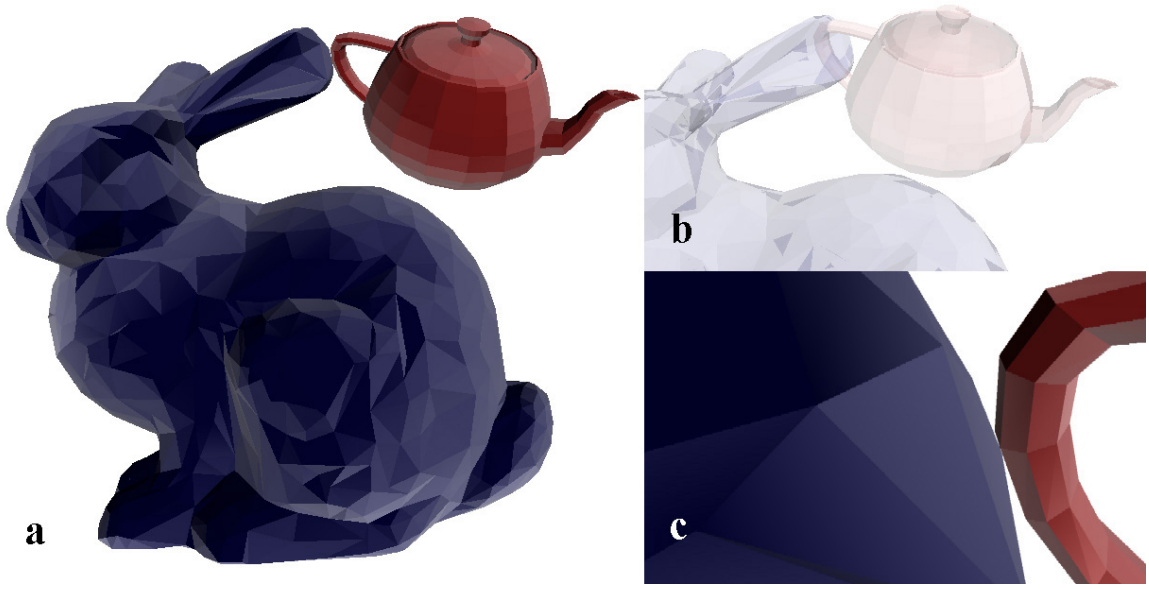


Fig. 2.9 – Manipulation d'objet. La détection de collisions continue permet bien d'éviter toute interpénétration lors de la manipulation. Une option de l'application permettait de visualiser la position qu'aurait eue l'objet si une méthode de détection de collisions discrète avait été employée (en b).

s'interpénétraient pas. L'utilisateur avait aussi la possibilité d'afficher la position finale intentionnelle de l'objet, c'est-à-dire la position que l'objet aurait eue si une méthode de détection de collisions discrète avait été employée.

La figure 2.9 montre une théière mobile, manipulée par l'utilisateur, qui vient d'entrer en contact avec un lapin statique. La partie (c) de l'image est un gros plan de la scène vue en (a), obtenu en se déplaçant dans la scène après le contact. La partie (b) montre la position finale intentionnelle de la théière. Les deux objets sont affichés en transparence, afin de mieux voir l'interpénétration qu'aurait entrainé une méthode de détection de collisions discrète.

La figure 2.10 montre une situation problématique pour une méthode de détection de collisions discrète, résolue par une méthode continue. En (a) et (b), le lapin a un mouvement général, comportant une translation de gauche à droite et une légère rotation. Les lapins transparents correspondent aux positions initiales et finales du lapin opaque. Puisque le mouvement ne dure qu'un seul pas de temps, une méthode discrète détecterait une collision dans le cas (a) seulement, et échouerait dans le cas (b). Dans les deux cas cependant, la méthode continue introduite dans ce chapitre parvient à stopper l'objet.

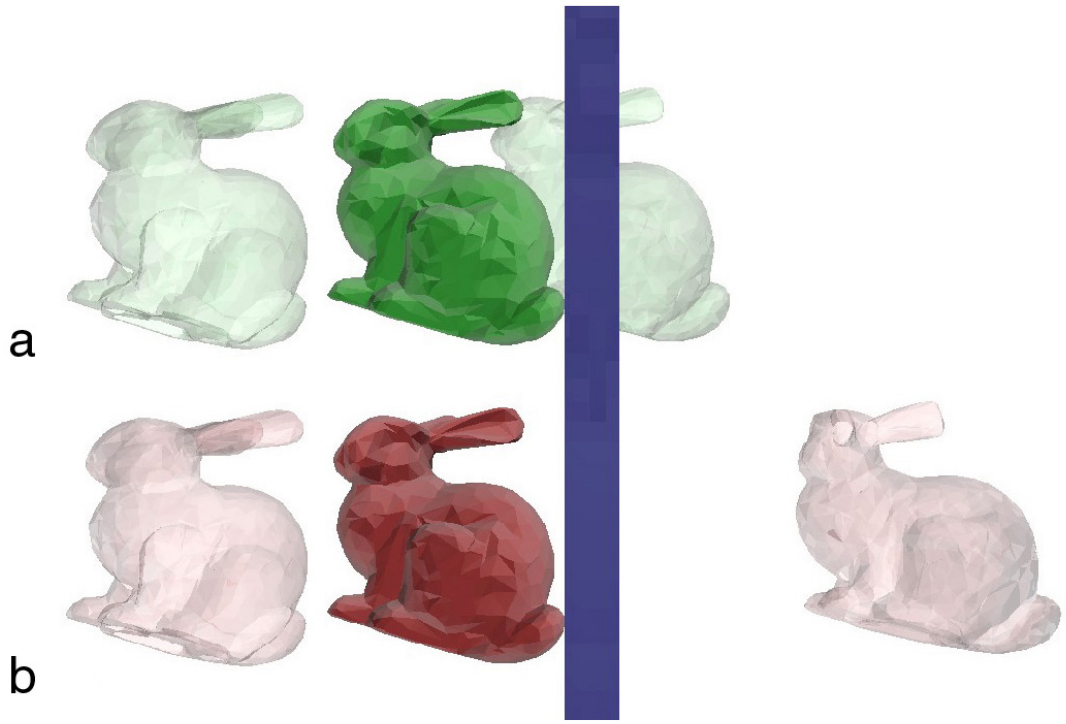


Fig. 2.10 – En b, une méthode discrète ne parvient pas à arrêter le lapin. La méthode continue introduite dans ce chapitre permet de détecter une collision dans les deux cas.

2.7.4 Inconvénients de cette première approche

Les tests effectués ont permis de valider l'utilisation d'un mouvement intermédiaire arbitraire pour détecter des collisions en continu efficacement sans perturber le réalisme des simulations dynamiques. En particulier, nous avons montré qu'un mouvement arbitraire spécifique, dérivé d'un vissage, permet de réduire les équations de détection de collisions élémentaires et les tests de recouvrement entre sphères englobantes à des polynômes de degré inférieur ou égal à trois. Ces polynômes, que l'on peut résoudre de façon simple et robuste, sont la clé de l'efficacité de cette première approche.

Cependant, nous avons pris pour hypothèse dans cette première approche que l'un des deux objets est immobile. Bien qu'il soit possible de détecter des collisions entre *deux* objets mobiles en considérant le mouvement relatif des objets, il est impossible de généraliser cette approche lorsque plus de deux objets sont mobiles. La raison essentielle de ce problème vient de ce qu'en composant deux mouvements arbitraires obtenus à partir des fonctions a et b spécifiques décrites par les équations 2.23 et 2.24, on n'obtient pas un mouvement arbitraire de même type. Il n'est donc pas possible de considérer les mouvements relatifs lorsque plus de deux objets sont mobiles.

Considérons en effet trois objets A, B et C. Pour détecter une collision entre

A et B, il faut calculer le mouvement relatif de A par rapport à B, et remplacer ce mouvement relatif par un vissage. De même, pour calculer une collision entre A et C, il faut calculer le mouvement relatif de A par rapport à C, et remplacer ce mouvement par un vissage. Cependant, si une collision intervient dans l'intervalle de temps considéré (et même si elle intervient entre d'autres objets que les objets A, B ou C), il est nécessaire de positionner les objets à l'instant de collision. Pour cela, puisque l'on a considéré le mouvement de A par rapport à B, il faut d'abord positionner B par le mouvement de vissage, puis positionner A en se servant du mouvement de vissage *relativement au mouvement de* B. Cependant, il faudrait de la même façon positionner A par rapport au mouvement de C. Il n'existe cependant aucune raison pour que les deux positions, celle obtenue à partir de A et celle obtenue à partir de B, correspondent.

Il existe bien sûr plusieurs applications qui ne nécessitent de déplacer qu'un seul objet à la fois, notamment certaines applications de prototypage virtuel. Cependant, nous souhaitons maintenant proposer une approche qui permette de détecter des collisions en continu lorsque plusieurs objets sont mobiles en même temps, en particulier lorsque certains objets sont des chaînes poly-articulées.

Un deuxième inconvénient réside dans le fait de prendre des sphères pour volumes englobants. Nous avons été incités à faire ce choix en raison de notre souhait, dans cette première approche, de réduire le *coût* des équations de détection de collisions élémentaires et des tests de recouvrement. Nous avons vu en effet qu'il était possible, en choisissant des sphères englobantes, de réduire également les tests de recouvrement continus à des polynômes de degré trois.

Les sphères cependant, sont une mauvaise approximation des objets peu sphériques ! Nous avons en effet pu constater que, dans certains cas, il est nécessaire de recouvrir une seule face d'un grand nombre de petites sphères (*cf* figure 2.6). Ceci peut réduire l'efficacité de la détection de collisions.

2.8 Conclusion

Dans ce deuxième chapitre, nous avons proposé d'utiliser un *mouvement intermédiaire arbitraire* (bien qu'obligatoirement rigide et continu), pour interpoler les positions successives des objets et résoudre efficacement les équations de détection de collisions continues. Nous avons ainsi montré que, loin d'être une restriction de la généralité de l'approche, un mouvement intermédiaire arbitraire permet de remplacer le mouvement de l'objet, généralement inconnu, sur des intervalles de temps successifs, *dans le même esprit que celui qui sous-tend la discrétisation des équations de la dynamique.*

L'aspect le plus important de la notion de mouvement arbitraire est donc sans doute la justification de son applicabilité, et non la proposition de tel ou tel mouvement particulier. Les méthodes continues préalables, en effet, proposaient déjà des mouvements intermédiaires spécifiques. Outre leur manque d'efficacité, pour les raisons que

nous avons exposées au chapitre précédent, leur recours à un mouvement particulier, généralement pour remplacer la trajectoire *globale* de l'objet et non sa trajectoire locale sur de petits intervalles de temps successifs, conduisait à les écarter au profit des méthodes discrètes [Bar93].

Nous avons également, dans une première approche, concentré nos efforts sur la réduction du coût des équations de détection de collisions, et avons pour cela proposé un premier type de mouvement arbitraire, dérivé d'un *vissage*, qui permet de réduire à la fois les tests élémentaires et les tests de recouvrement continus entre sphères englobantes à des polynômes de degré inférieur ou égal à trois, qui peuvent être résolus simplement et efficacement par les méthodes de Cardan.

Nous avons présenté plusieurs optimisations de la détection de collisions et montré, en particulier, que les inconvénients liés à l'utilisation de sphères comme volumes englobants pouvaient être diminués par une *méthode d'étiquetage*.

Plusieurs tests ont permis de valider l'utilisation d'un mouvement intermédiaire arbitraire pour détecter des collisions en continu efficacement sans perturber le réalisme des simulations dynamiques.

Nous avons cependant remarqué qu'en nous concentrant sur le *coût* des équations de détection de collisions, nous avions en quelque sorte été incités à utiliser des sphères, peu efficaces lorsque les objets sont fins ou étroits.

Dans le chapitre suivant, nous allons présenter une deuxième approche, toujours fondée sur l'utilisation de mouvements intermédiaires arbitraires, dans laquelle nous allons nous attacher à réduire le *nombre* d'équations de détection de collisions, en choisissant des volumes englobants plus adaptés et en utilisant des méthodes de résolution adaptées à ces volumes englobants. En ne cherchant plus à obtenir des équations de détection de collisions polynômiales, nous allons également pouvoir détecter des collisions entre plusieurs objets mobiles.

Chapitre 3

Arithmétique d'intervalles et boîtes englobantes orientées

Ce chapitre présente une deuxième approche envisagée pour détecter des collisions en continu entre objets polyédriques rigides. Contrairement à la première approche, la méthode proposée dans ce chapitre permet de gérer le cas d'objets simultanément mobiles, et celui des chaînes polyarticulées rigides. Nous montrons aussi comment la méthode introduite ici peut accélérer les méthodes connues de détection de collisions continues entre surfaces paramétriques ou implicites rigides.

3.1 Introduction

Le chapitre précédent s'est concentré sur la définition de la notion de mouvement intermédiaire arbitraire rigide, et sur son application à la détection de collisions continue entre objets polyédriques rigides. Dans une première approche, nous nous sommes attachés à définir une classe de mouvements particuliers, dérivés d'un vissage, qui permet de réduire à la fois les équations de détection de collisions élémentaires (arête/arête, point/face et face/point), et les tests continus de recouvrement entre sphères, à des polynômes de degré trois. Ces polynômes, dont les coefficients peuvent être calculés à l'aide des coordonnées des primitives et des paramètres du mouvement de l'objet, peuvent être résolus simplement à l'aide des formules de Cardan. Ces polynômes, ainsi que le principe sous-jacent de l'utilisation d'un mouvement arbitraire, sont la clé de l'efficacité de cette première approche.

Une hypothèse importante de la première approche, cependant, est que l'un des deux objets est immobile. Bien qu'il soit possible de considérer le mouvement relatif de deux objets pour effectuer une détection de collisions entre deux objets mobiles, il n'est pas possible d'utiliser la classe de mouvements choisis dès que plus de deux objets sont simultanément en contact, ou s'apprêtent à l'être. La loi de composition, en effet, n'est pas stable sur la classe de mouvements intermédiaires choisis. Par ailleurs, la contrainte

Fig. 3.1 – *Positionnement précis d'une portière de voiture.* La méthode de détection de collisions continue décrite dans ce chapitre permet de positionner la portière précisément (sans interpénétrations) et interactivement. Le squelette de la voiture comporte environ 29000 triangles. La portière en contient environ 16000 (modèles 3d ©Renault).

que nous nous sommes imposés en souhaitant obtenir des équations de détection de collisions que l'on peut résoudre exactement (comme c'est le cas des polynômes de degré trois), nous a conduit à utiliser des hierarchies de sphères pour accélérer la détection de collisions. Nous avons remarqué cependant que, dans le cas d'objets fins ou plats, il est parfois nécessaire d'utiliser de nombreuses petites sphères pour recouvrir l'objet. Afin d'éviter un ralentissement important de la détection de collisions, nous avons été amenés à développer une optimisation spécifique, la méthode des étiquettes. Bien qu'efficace lorsque l'objet présente une importante cohérence spatiale, cette optimisation peut s'avérer inutile dès lors que l'objet comporte, par exemple, des surfaces rugueuses, ou même simplement triangulées.

Dans une deuxième approche, décrite dans ce chapitre, nous souhaitons concentrer nos efforts sur l'étude de volumes englobants plus appropriés aux objets poly-

édriques. Les tests de recouvrement entre volumes englobants représentent en effet une part non négligeable du temps total de détection de collisions entre deux objets, comme indiqué par la traditionnelle équation de coût (équation (1.5)). Aussi, nous avons supprimé la contrainte qui consiste à chercher à obtenir des équations simples (*i.e.* que l'on peut résoudre analytiquement), qui entravait trop le choix du volume englobant, et avons cherché à obtenir un test *continu* de recouvrement entre volumes englobants le plus efficace possible.

Nous avons choisi d'avoir recours à des hiérarchies de boîtes englobantes orientées (OBBs) en raison de leurs performances dans un grand nombre d'applications, en particulier dans celles ou les objets peuvent être très proches les uns des autres. Les hiérarchies sont des arbres binaires construit classiquement [Got99, GLM96]. Nous avons pu constater, avec les auteurs précédents, que les hiérarchies construites à partir d'une approche *top-down* les mieux ajustées sont généralement obtenues grâce à la règle de répartition *min-max* [Got99]. Dans une approche *top-down*, la hiérarchie est construite de haut en bas : pour un ensemble de triangles donnés, une boîte englobante est construite, puis les triangles sont répartis dans deux sous-ensembles et le processus est itéré tant qu'un sous-ensmelbe contient plus d'un triangle. Essentiellement, la règle de répartition *min-max* consiste à choisir les sous-ensembles de triangles de façon à minimiser la taille de la plus grande des deux boîtes englobant les sous-ensembles. La figure 3.2 montre plusieurs niveaux de la hiérarchie de boîtes englobantes orientées associée à une théière.

Le test discret de recouvrement entre deux OBBs statiques le plus efficace est probablement celui décrit par Gottschalk *et al.* [GLM96], qui repose sur le *théorème de l'axe séparateur*.

Supposons que le premier OBB soit décrit par trois axes $\mathbf{e}_1$, $\mathbf{e}_2$ et $\mathbf{e}_3$, un centre $\mathbf{T}_A$, et ses demi-tailles suivant ses axes a_1, a_2 et a_3. De la même façon, le second OBB est décrit par ses axes $\mathbf{f}_1$, $\mathbf{f}_2$ et $\mathbf{f}_3$, son centre $\mathbf{T}_B$, et ses demi-tailles suivant ses axes b_1, b_2 et b_3. D'après le théorème de l'axe séparateur, deux OBBs statiques se recouvrent si et seulement si quinze tests d'axe séparateur échouent. Un test d'axe séparateur est simple : l'axe $\mathbf{a}$ sépare les OBBs si et seulement si

$$|\mathbf{a} \cdot \mathbf{T}_A\mathbf{T}_B| > \sum_{i=1}^{3} a_i|\mathbf{a} \cdot \mathbf{e}_i| + \sum_{i=1}^{3} b_i|\mathbf{a} \cdot \mathbf{f}_i| \tag{3.1}$$

Les quinzes axes suffisants sont déduits des axes des OBBs :

$$\mathbf{a} \in \{\mathbf{e}_i,\ \mathbf{f}_j,\ \mathbf{e}_i \wedge \mathbf{f}_j,\ 1 \leqslant i \leqslant 3,\ 1 \leqslant j \leqslant 3\} \tag{3.2}$$

Les termes de l'inégalité (3.1) ont une interprétation géométrique simple : le membre de gauche correspond à la distance entre les centres, suivant la direction de $\mathbf{a}$, et le membre de gauche correspond à la somme des demi-rayons des boîtes, suivant la même direction. La figure 3.3 montre un exemple d'application dans le plan. Dans cet

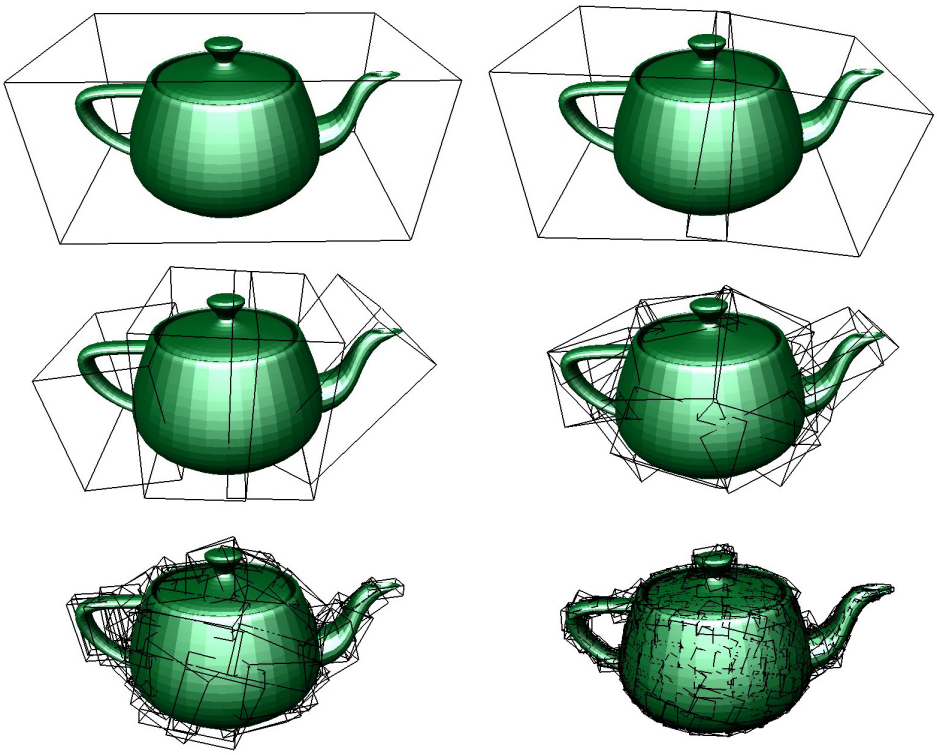


Fig. 3.2 – Niveaux 1, 2, 3, 5, 7 et 10 de la hiérarchie de boîtes englobantes associée à la théière.

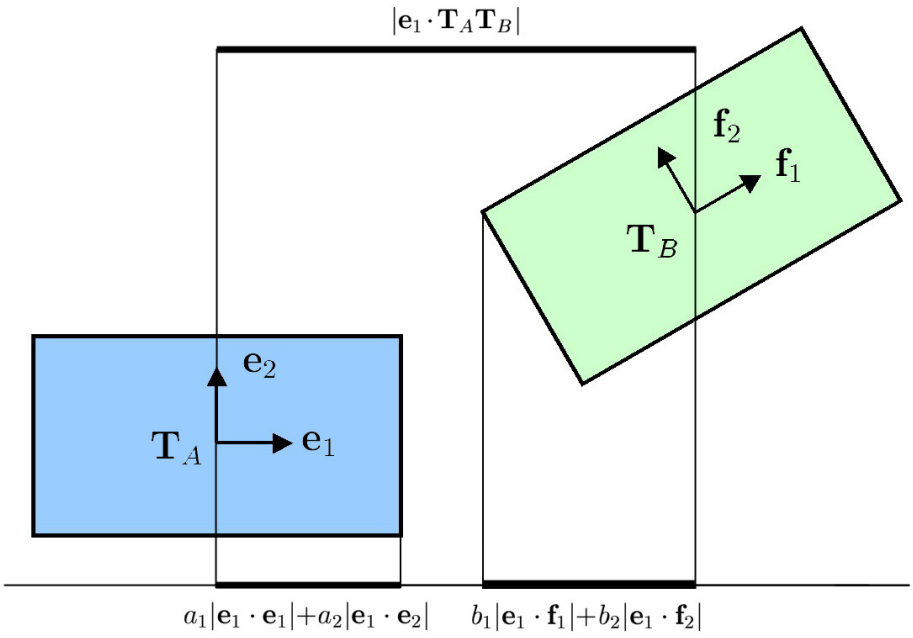


Fig. 3.3 – L'axe $\mathbf{e}_1$ sépare les deux boîtes orientées car, dans la direction de l'axe, la distance entre les centres des boîtes $|\mathbf{e}_1 \cdot \mathbf{T}_A\mathbf{T}_B|$ est plus grande que la somme de leurs rayons, égale à $(a_1|\mathbf{e}_1 \cdot \mathbf{e}_1| + a_2|\mathbf{e}_1 \cdot \mathbf{e}_2|) + (b_1|\mathbf{e}_1 \cdot \mathbf{f}_1| + b_2|\mathbf{e}_1 \cdot \mathbf{f}_2|)$.

exemple, l'axe $\mathbf{e}_1$ est séparateur, car la distance entre les centres dans cette direction, égale à $|\mathbf{e}_1 \cdot \mathbf{T}_A\mathbf{T}_B|$, est supérieure à la somme des deux rayons apparents (*i.e.* dans la direction de $\mathbf{e}_1$), égale à $(a_1|\mathbf{e}_1 \cdot \mathbf{e}_1| + a_2|\mathbf{e}_1 \cdot \mathbf{e}_2|) + (b_1|\mathbf{e}_1 \cdot \mathbf{f}_1| + b_2|\mathbf{e}_1 \cdot \mathbf{f}_2|)$.

Il est formellement simple de transformer ce test discret en un test continu. En effet, lorsque les boîtes englobantes se déplacent au cours d'un intervalle de temps, les deux membres de l'inégalité (3.1) sont des fonctions continues du temps. Cependant, en raison notamment des valeurs absolues intervenant dans les tests d'axe séparateur (3.1), le mouvement intermédiaire décrit dans les chapitres précédents 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. C'est la raison pour laquelle nous avons préféré abandonner la recherche d'équations polynômiales simples.

Notons que nous ne perdons pas de vue l'esprit de la première approche. Le(s) mouvement(s) intermédiaire(s) que nous proposons d'utiliser dans ce chapitre sont encore simples, en ce sens qu'ils favorisent une implantation efficace de la détection de collisions. Cependant, nous ne cherchons plus à obtenir des solutions algébriques aux équations de détection de collisions. De plus, nous évitons ainsi le problème dû à la composition des mouvements, et pouvons donc détecter des collisions entre plusieurs objets mobiles et des chaînes poly-articulées acycliques rigides. Nous avons vu précédemment qu'un mouvement de vissage rigide, de paramètres ω et s, peut être décrit de la manière suivante :

$$\mathbf{V}(t) = \begin{pmatrix} \cos(\omega.t) & -\sin(\omega.t) & 0 & 0 \\ \sin(\omega.t) & \cos(\omega.t) & 0 & 0 \\ 0 & 0 & 1 & s.t \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.3}$$

Dans ce cas, la rotation et la translation au cours du vissage sont constantes. Alors que ce mouvement entraîne un système transcendant inacceptable dans l'approche précédente car impossible à résoudre exactement, nous pouvons maintenant l'utiliser. De même, nous pouvons tout aussi bien remplacer les fonctions *cosinus* et *sinus* par toute paire de fonctions $C : [0,1] \to \mathbb{R}$ et $S : [0,1] \to \mathbb{R}$ satisfaisant :

$$C^2(t) + S^2(t) = 1 \quad t \in [0,1] \tag{3.4}$$

et les conditions initiales et finales :

$$\begin{cases} C(0) = 1 \\ C(1) = cos(\omega) \\ S(0) = 1 \\ S(1) = sin(\omega) \end{cases} \tag{3.5}$$

afin d'optimiser au mieux l'implantation des fonctions de détection de collisions.

S'il s'avère par exemple que l'évaluation d'une racine carrée est moins coûteuse que celle de fonctions trigonométriques, nous pouvons contraindre ω à appartenir à

l'intervalle $[0, \frac{\pi}{2}]$ et prendre :

$$\begin{cases} C(t) = cos(\omega).t \\ S(t) = \sqrt{1 - C^2(t)} \end{cases} \tag{3.6}$$

Le point important est que nous pouvons maintenant composer des mouvements du type (2.7) et donc détecter des collisions entre plusieurs objets mobiles et/ou des chaines poly-articulées acycliques[1] rigides.

Aussi, afin de résoudre les équations élémentaires d'une part, et de détecter continûment des recouvrements entre volumes englobants d'autre part, nous nous reposons maintenant sur *l'arithmétique d'intervalles*. Plus précisément, les algorithmes décrits dans ce chapitre reposent sur l'intégration efficiente de l'arithmétique des intervalles (AI) et des hiérarchies de boîtes englobantes orientées (OBBs). Les deux approches bénéficient l'une de l'autre. 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 OBBs mobiles à partir du test de recouvrement discret.

Inversement, les boîtes englobantes permettent, à l'instar des sphères du chapitre précédent, 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.

Rappelons en effet, comme cela a été vu dans le premier chapitre, que l'arithmétique d'intervalles a déjà été utilisée en détection de collisions continue [VHBZ90, Duf92, SWFCB93]. Cependant, ces méthodes n'utilisent pas de tests de recouvrements continus entre boîtes englobantes orientées pour accélérer la détection de collisions, et ne sont pas utilisables ou facilement adaptables pour détecter des collisions entre objets polyédriques complexes [Sny95].

La prochaine section présente les principes de base de l'arithmétique d'intervalles. La section 3.3 introduit le test de recouvrement continu entre deux OBBs et présente les tests de détection de collisions élémentaires (arête/arête, point/face ou face/point). La section 3.4 explique brièvement comment la méthode introduite dans ce chapitre peut être utilisée pour accélerer les techniques de détection de collisions entre surfaces paramétriques ou implicites rigides. La section 3.5 présente plusieurs optimisations de la méthode, permettant d'accélérer significativement la détection de collisions. Enfin, la section 3.6 conclut le chapitre.

[1] Acycliques, car une boucle créerait une incohérence dans les mouvements des objets de la chaîne, comme dans l'approche précédente. Ce qui n'est pas génant pour un groupe d'objet en contact, en raison de la distance de sécurité entre les objets (*cf* chapitre 6), devient un problème pour des objets dont les mouvements sont continûment dépendant les uns des autres au cours de l'intervalle de temps.

3.2 Arithmétique d'intervalles

En quelques mots, l'arithmétique d'intervalles consiste à calculer avec des intervalles réels plutôt qu'avec des nombres réels. Sous sa forme moderne, elle a été introduite par le travail de thèse de R.E. Moore [Moo62] et est désormais un sujet de recherche à part entière. Une introduction générale à l'arithmétique d'intervalles est donnée par exemple par Kearfott [Kea96]. Snyder [Sny92] a présenté quelques applications de l'aritmétique d'intervalles à l'informatique graphique, notamment pour obtenir des approximations de courbes implicites.

Nous utiliserons seulement des intervalles fermés. Pour mémoire, la définition d'un intervalle réel fermé $[a, b]$ est :

$$I = [a, b] = \{x \in \mathbb{R},\ a \leqslant x \leqslant b\}$$

Cette définition peut être généralisée aux vecteurs d'éléments de $\mathbb{R}$:

$$\begin{aligned} I_n &= [a_1, b_1] \times ... \times [a_n, b_n] \\ &= \{\mathbf{x} = (x_1,...,x_n) \in \mathbb{R}^n,\ a_i \leqslant x_i \leqslant b_i \quad \forall i,\ \ 1 \leqslant i \leqslant n\} \end{aligned}$$

L'ensemble des intervalles de nombres réels est noté $\mathbb{IR}$, tandis que l'ensemble des intervalles vectoriels est noté $\mathbb{IR}^n$.

Les opérations élémentaires sur les nombres réels peuvent être transposées aux intervalles :

$$\begin{aligned} [a, b] + [c, d] &= [a + c, b + d] \\ [a, b] - [c, d] &= [a - d, b - c] \\ [a, b] \times [c, d] &= [min(ac, ad, bc, bd), max(ac, ad, bc, bd)] \\ 1/[a, b] &= [1/b, 1/a] \quad \text{si } a > 0 \text{ ou } b < 0 \\ [a, b]/[c, d] &= [a, b] \times (1/[c, d]) \quad \text{si } c > 0 \text{ ou } d < 0 \\ [a, b] &\leqslant [c, d] \quad \text{si } b \leqslant c \end{aligned} \tag{3.7}$$

Pour les opérations dans $\mathbb{IR}^n$, les calculs sont effectuées coordonnée par coordonnée.

L'arithmétique des intervalles permet de borner naturellement des fonctions de la variable réelle sur des intervalles. Précisément, à toute fonction réelle donnée $f : \mathbb{R} \to \mathbb{R}$, on associe une *fonction d'inclusion* $\tilde{f} : \mathbb{IR} \to \mathbb{IR}$ telle que

$$x \in I \Rightarrow f(x) \in \tilde{f}(I) \tag{3.8}$$

pour tout intervalle I de $\mathbb{R}$.

Les fonctions d'inclusion idéales sont celles qui bornent exactement l'intervalle image $f(I)$, quel que soit l'intervalle I. S'il est parfois difficile de calculerdes fonctions d'inclusions idéales, il est beaucoup plus facile de calculer des fonctions d'inclusion

pratiques, *i.e.* qui suffisent à résoudre notre problème, en utilisant l'arithmétique d'intervalles.

Ainsi, d'après les équations (2.7) et (3.3), les deux fonctions d'inclusion dont nous avons besoin sont celles des fonctions *sinus* et *cosinus*. Une seule autre fonction d'inclusion est requise dans le test continu de recouvrement entre boîtes englobantes introduit dans la section 3.3, la fonction *valeur absolue*. Pour ces fonctions, des fonctions d'inclusion peuvent être obtenues facilement, comme expliqué dans Snyder [Sny92]. Ensuite, les opérations élémentaires (equation (3.7)) sont récursivement appliquées pour calculer les fonctions d'inclusion des coordonnées d'un point ou d'un vecteur.

Considérons par exemple le cas d'un point mobile dans le repère global $\mathcal{R}_0$. Nous avons vu au chapitre précédent que ses coordonnées sont :

$$\mathbf{x}_G(t) = \mathbf{P}^{-1}\mathbf{V}(t)\mathbf{P}\mathbf{P}_o\mathbf{x}_o \tag{3.9}$$

où $\mathbf{x}_o$ représente les coordonées du point dans le repère de l'objet, et $\mathbf{P}_o$ est la matrice de passage du repère de l'objet au repère global $\mathcal{R}_0$.

Par conséquent, les fonctions d'inclusion des coordonnées sont :

$$\tilde{\mathbf{x}}_G(t) = \mathbf{P}^{-1}\tilde{\mathbf{V}}(t)\mathbf{P}\mathbf{P}_o\mathbf{x}_o \tag{3.10}$$

où les opérations sont effectuées sur les intervalles, comme indiqué dans l'équation (3.7).

L'intérêt des fonctions d'inclusions est qu'elles fournissent un moyen simple de calculer de façon robuste les racines d'une fonction $f : I \to \mathbb{R}$. La plus simple de ces méthodes est celle de la subdivision binaire récursive.

Supposons qu'une fonction d'inclusion $\tilde{f}$ soit disponible. Alors $\tilde{f}(I) = [a, b]$ est calculé. Si $a > 0$ ou $b < 0$, il ne peut y avoir de racine dans l'intervalle I. Dans le cas contraire, il *peut* y avoir une racine (mais pas nécessairement, puisque $\tilde{f}(I)$ peut ne pas borner exactement l'image de f sur I, et/ou que f peut ne pas être continue). Dans ce cas, l'intervalle I est divisé en deux intervalles égaux $[a, m]$ et $[m, b]$, où $m = 1/2(a+b)$, et les calculs sont maintenant effectués sur les deux intervalles plus petits. Ce processus se poursuit récursivement jusqu'à ce que la taille d'un intervalle I soit plus petite qu'un seuil pré-déterminé. Dans ce cas, l'algorithme déclare qu'une racine a été trouvée. Il peut être montré que si f est continue sur l'intervalle I, l'algorithme ne peut manquer aucune racine. C'est le cas dans le problème qui nous intéresse puisque les mouvements des objets sont continus.

Dans le cas de la détection de collision, nous verrons que le seuil imposé sur I permet d'imposer une précision à la fois sur l'instant de premier contact, et sur la distance parcourue par les primitives durant l'intervalle de temps I (*cf* section 3.3).

Remarquons cependant que si la fonction f est nulle sur un intervalle non vide I_0, cette méthode de calcul de racines ne renvoie qu'un nombre fini d'intervalles, dont l'union contient I_0, et dont le nombre dépend du seuil pré-déterminé.

Notons enfin que l'arithmétique d'intervalles permet de gérer naturellement le problème de la précision finie d'un ordinateur. Les opérations (3.7) supposent en effet une précision infinie. Sur un ordinateur conforme aux normes IEEE de calcul flottant cependant, il est possible d'effectuer les calculs en arrondissant par défaut pour les bornes inférieures, et par excès pour les bornes supérieures. Dans notre implantation, les valeurs de tolérance sont supérieures à la précision du processeur, et il est possible de ne pas se soucier de ce niveau de précision sans entraver la robustesse de la détection de collisions.

3.3 Arithmétique d'intervalles et détection de collisions continue

Nous pouvons maintenant décrire l'intégration des algorithmes de détection de collisions. Nous dérivons le test continu de recouvrement entre OBBs à partir du test discret proposé par Gottschalk *et al.* [GLM96], et nous détaillons les méthodes de détection de collisions élémentaires entre les primitives polyédriques (sommets, arêtes et faces). Enfin, nous nous intéressons au problème de la précision de la détection de collision, naturellement pris en compte par l'arithmétique d'intervalles.

3.3.1 détection de collisions entre OBBs

Puisque nous souhaitons détecter des collisions en continu, il nous faut un moyen de déterminer si deux OBBs mobiles se recouvrent *durant* un intervalle de temps $[t_0, t_1]$, et pas seulement aux positions initiales et finales.

Un point important est que nous n'avons besoin que d'un test *conservatif*. En effet, bien qu'un recouvrement entre deux OBBs *doive* être détecté, ce n'est pas fondamentalement un problème de déclarer *à tort* qu'il y en a eu un. Lorsque l'algorithme de traversée des hiérarchies englobantes se termine, aucune collision ne peut avoir été manquée. Notons que des tests conservatifs sont utilisés dans d'autres méthodes de détection de collision, notamment celles faisant intervenir les k-dops [KHMSZ98].

Le test continu de recouvrement est constitué de deux étapes :

1. Effectuer d'une version continue des quinze tests d'axe séparateur.
2. Si la première étape conclut que les OBBs 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.

3.3.1.1 Tests d'axes séparateurs continus

La première étape est obtenue facilement à partir du test discret de recouvrement et de l'arithmétique d'intervalles. Les deux membres de l'inégalité (3.1) sont des fonc-

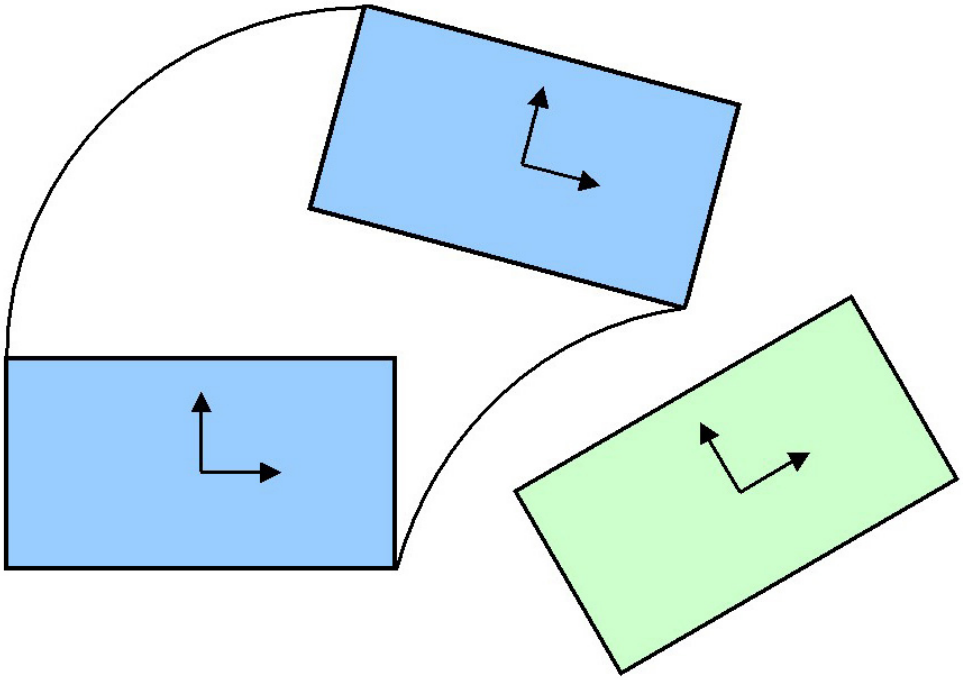


Fig. 3.4 – Aucun axe ne permet de séparer les boîtes durant tout l'intervalle de temps.

tions continues du temps dépendant des positions (changeantes) des OBBs au cours de l'intervalle de temps. Les deux fonctions peuvent être bornées en utilisant l'arithmétique d'intervalles, comme décrit dans la section 3.2.

Notons $[l_1, l_2]$ un encadrement du membre de gauche de l'inégalité (3.1), et $[r_1, r_2]$ un encadrement du membre de droite de la même inégalité. Si $l_1 > r_2$, l'axe $\mathbf{a}$ sépare les deux OBBs sur tout l'intervalle de temps. La première étape consiste ainsi à effectuer quinze tests d'axes séparateurs similaires.

Cette première étape, cependant, peut seulement détecter un axe qui sépare les deux OBBs *durant l'intervalle de temps tout entier*. Pourtant, les deux OBBs peuvent ne pas s'interpénétrer pendant l'intervalle de temps et être séparés par des axes différents au cours du mouvement. Ceci ne peut pas être détecté par la première étape du test continu : les quinze tests d'axe séparateurs continus échouent, et la première étape de l'algorithme déclare que les OBBs se recouvrent au cours de l'intervalle de temps. Ce phénomène est représenté dans la figure 3.4. Aucun des axes (quatre dans le cas de boîtes planes) n'est séparateur durant tout l'intervalle de temps.

Dans les méthodes de recherche de racines utilisant l'arithmétique d'intervalles, ce problème est résolu en subdivisant l'intervalle, ou en recourant à des méthodes plus sophistiquées pour réduire la taille de l'intervalle, comme les méthodes de Newton par intervalles [Sny92, SWFCB93]. Afin d'éviter des subdivisions automatiques ou des méthodes gourmandes en temps de calcul, nous proposons d'utiliser comme seconde étape un simple *test de subdivision*. Ce test est une heuristique, dans le même esprit

que les méthodes de Newton par intervalles, puisqu'il dépend des vitesses des OBBs, mais est beaucoup moins coûteux en calculs.

3.3.1.2 Test de subdivision

Généralement, un axe séparant les OBBs à un instant donné ne les séparera pas durant l'intervalle de temps entier si les objets bougent trop vite *relativement à leur taille*. Ceci est précisément ce que le test de subdivision vérifie : brièvement, les OBBs sont projetés sur la direction de leur vitesse relative. En premier lieu, la vitesse relative des centres des OBBs au début de l'intervalle de temps est calculée (ceci revient à considérer que le deuxième OBB est immobile).

En supposant que $\mathbf{T}_A$ est exprimée dans le repère global $\mathcal{R}_0$, alors sa vitesse peut être déduite de l'équation (3.9) :

$$\mathbf{v}(\mathbf{T}_A) = \mathbf{P}_A^{-1}\mathbf{V}'_A(t_0)\mathbf{P}_A\mathbf{T}_A \tag{3.11}$$

où t_0 est la borne inférieure de l'intervalle de temps courant, et où $\mathbf{P}_A$ et $\mathbf{V}_A(t_0)$ décrivent le vissage associé au premier objet. La matrice $\mathbf{V}'_A(t_0)$ est calculée simplement à partir de l'équation (3.3) en différentiant les éléments de la matrice par rapport au temps.

$$\mathbf{V}'_A(t_0) = \begin{pmatrix} -\omega_A \sin(\omega_A.t_0) & -\omega_A \cos(\omega_A.t_0) & 0 & 0 \\ \omega_A \cos(\omega_A.t_0) & -\omega_A \sin(\omega_A.t_0) & 0 & 0 \\ 0 & 0 & 0 & s_A \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.12}$$

Des expressions semblables peuvent être déduites pour le second OBB. Notons maintenant $\mathbf{v}_r = \mathbf{v}(\mathbf{T}_A) - \mathbf{v}(\mathbf{T}_B)$ la vitesse relative des centres des OBBs. Alors $(t_1 - t_0)|\mathbf{v}_r|$ est approximativement la taille du chemin relatif parcouru par les centres des OBBs durant l'intervalle de temps. Ainsi, l'intervalle de temps $[t_0, t_1]$ est subdivisé si et seulement si :

$$\sum_{i=1}^{3} a_i |\mathbf{v}_r \cdot \mathbf{e}_i(t_0)| + (t_1 - t_0)|\mathbf{v}_r| > k. \sum_{i=1}^{3} b_i |\mathbf{v}_r \cdot \mathbf{f}_i(t_0)| \tag{3.13}$$

où k est une constante pré-déterminée. Si l'intervalle n'est pas subdivisé, l'algorithme déclare que les OBBs se sont interpénétrés durant l'intervalle de temps. C'est ici que le test continu de recouvrement n'est que conservatif. L'algorithme peut déclarer à tort que deux OBBs se recouvrent durant un intervalle de temps. Cependant, il ne manque aucun recouvrement réel. De plus, nos expériences ont montré que la plupart des fausses déclarations peuvent être éliminées en prenant $k = 0.15$ (*cf* section 3.5.2).

3.3.2 détection de collisions entre primitives polyédriques

Comme nous l'avons rappelé dans la première approche, la détection de collisions continue entre objets polyédriques est en quelque sorte plus simple, bien que plus

coûteuse, que la détection de collisions discrète, puisque toutes les configurations de contact impliquent l'un au moins des trois types de contacts non dégénérés : arête/arête, point/face et face/point. De plus, ainsi que nous l'avons vu au chapitre précédent, chaque test peut être formulé de façon que le premier instant de contact soit une racine d'une fonction du temps.

Dans le cas arête/arête, une collision est d'abord détectée entre *les droites* contenant les arêtes en résolvant l'équation (1.3) grâce à la méthode de recherche de racines par intervalles décrite dans la section 3.2. Une solution dans $[0, 1]$ est conservée si et seulement si le point d'intersection des droites appartient aux arêtes.

Dans les cas point/face et face/point, une collision est d'abord détectée entre le point et *le plan* contenant la face en résolvant l'équation (1.4) grâce à la même méthode de résolution par intervalles. Les solutions t_r dans $[0, 1]$ sont conservées si et seulement si $\mathbf{a}(t_r)$ est à l'intérieur du triangle $\mathbf{b}(t_r)\mathbf{c}(t_r)\mathbf{d}(t_r)$.

Remarquons que dans la méthode de résolution récursive décrite dans la section 3.2, lorsqu'un intervalle de temps $[t_l, t_r]$ est divisé en deux intervalles $[t_l, t_m]$ et $[t_m, t_r]$, les solutions sont d'abord récursivement cherchées dans $[t_l, t_m]$. Si *et seulement si* aucune solution valide n'est trouvée dans cet intervalle, alors le second intervalle $[t_m, t_r]$ est récursivement examiné. Ceci permet de faire en sorte que la première racine valide obtenue est celle correspondant au premier contact entre les objets. De plus, puisque la méthode de résolution de la section 3.2 renvoie des intervalles, la valeur utilisée pour estimer l'instant de première collision est la borne inférieure de l'intervalle, offrant ainsi une sécurité supplémentaire contre les erreurs d'arrondi.

3.3.3 Précision de la détection de collisions

La méthode de résolution utilisée dans les tests entre primitives dépend d'un seuil sur la taille maximale d'un intervalle I_c qui peut être renvoyé par une fonction de détection de collisions entre primitives. Cette taille maximale définit clairement la précision sur l'instant de premier contact, puisque c'est un intervalle de temps. Il est ainsi possible de définir naturellement une première borne sur l'intervalle renvoyé I_c :

$$I_c < \epsilon_0 \tag{3.14}$$

Il est en fait aussi possible d'imposer une précision sur les positions des primitives à l'instant de premier contact renvoyé par la fonction de détection de collisions en définissant une autre taille maximale ϵ_1 pour le dernier intervalle I_c renvoyé. La taille maximale effectivement autorisée est ensuite définie comme étant la plus petite des deux bornes ϵ_0 et ϵ_1.

La borne imposée par la tolérance sur la position des primitives s'obtient simplement en utilisant le mouvement des objets. Utilisons par exemple le mouvement de vissage simple, à vitesses de rotation et de translation constantes, donné par l'équation

(3.3). Si dt est un petit intervalle de temps, alors la taille dl du chemin hélicoïdal suivi par un point $\mathbf{p}$ d'un objet suivant ce vissage est :

$$\begin{aligned} dl &= |\mathbf{p}'(t)|dt \\ &= \sqrt{s^2 + \omega^2(x^2 + y^2)}dt \end{aligned} \tag{3.15}$$

où s et ω sont les paramètres du vissage associé à l'objet, et où x et y sont les deux premières coordonnées de $\mathbf{p}$ dans le repère local du vissage. Notons ϵ_p l'erreur maximale autorisée sur la position du point $\mathbf{p}$ lorsque la méthode de subdivision se termine. Nécessairement :

$$dl < \epsilon_p \tag{3.16}$$

pour le dernier intervalle de temps renvoyé. Afin d'imposer ceci, un intervalle de temps valide (*i.e.* qui contient une racine) doit être subdivisé jusqu'à ce que

$$dt < \epsilon 1 = \frac{\epsilon_p}{\sqrt{s^2 + \omega^2(x^2 + y^2)}} \tag{3.17}$$

Puisque les positions des points de contact ne peuvent être connues à l'avance, une borne semblable est calculée pour chacun des quatre points $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ et $\mathbf{d}$ définissant les primitives polyédriques du test élémentaire courant (*cf* les équations pour le test arête/arête (1.3) et le test point/face (1.4)). La taille maximale de l'intervalle est ensuite définie comme le plus petit de ces quatre majorants. Puisque les primitives polyédriques (c'est-à-dire les sommets, les arêtes et les faces) sont convexes, ceci permet d'assurer que l'erreur maximale autorisée sur la position du point de contact est respectée.

Pour un autre mouvement intermédiaire, tel celui donné en introduction de ce chapitre (équation (3.6)), les bornes s'obtiennent de façon similaire.

3.4 Extension aux surfaces paramétrées ou implicites

Utiliser l'arithmétique d'intervalles pour détecter des collisions entre surfaces paramétrées ou implicites, comme le font Snyder *et al.* [SWFCB93] revient à calculer *au moment de la détection* des boîtes englobantes alignées sur les axes du repère global.

Considérons par exemple la surface paramétrée mobile (et éventuellement déformable) suivante :

$$\mathcal{P} : \begin{cases} \mathbb{R}^4 \to \mathbb{R}^3 \\ (\mathbf{x}, t) \mapsto \mathbf{y} = \mathcal{P}(\mathbf{x}, t) \end{cases} \tag{3.18}$$

et notons $X \times I$ un intervalle dans $\mathbb{IR}^3 \times \mathbb{IR}$. D'après la définition d'une fonction d'inclusion, $Y := \tilde{\mathcal{P}}(X)$ est un intervalle dans $\mathbb{IR}^3$. Géométriquement, Y est donc une boîte englobante alignée sur les axes du repère global. Plus précisément, cette boîte englobe la partie de l'objet paramétrée par X au cours de l'intervalle de temps I. Pour détecter une interpénétration ou un premier contact entre deux surfaces paramétrées,

il suffit donc d'évaluer Y sur des intervalles $X \times I$ de plus en plus petits pour chaque surface, comme lors de l'utilisation de hiérarchies de volumes englobants.

Cependant, Snyder *et al.* remarquent qu'évaluer Y peut être extrêmement coûteux pour des objets complexes, puisque la structure décrivant l'objet (ou sa fonction d'inclusion) doit être traversée.

Pour des surfaces paramétrées ou implicites *rigides*, une hiérarchie de boîtes orientées englobantes peut être précalculée et le test continu de recouvrement d'OBBs introduit dans cette section peut être utilisé pour éliminer rapidement des tests non pertinents. En effet, quelle que soit la complexité sous-jacente de la partie de l'objet englobée, le test de recouvrement entre OBBs prend un temps constant (contrairement à l'évaluation de $Y = \tilde{\mathcal{P}}(X)$). C'est lorsque des feuilles de la hiérarchie englobante se recouvrent, seulement, que les tests entre les parties des objets contenues dans ces feuilles peuvent être effectués grâce à la méthode de Snyder *et al.*

3.5 Optimisations

Comme dans l'approche précédente, il est possible de développer des optimisations spécifiques à l'utilisation de boîtes englobantes en continu. Afin de tester ces optimisations, nous avons de nouveau couplé les algorithmes de détection de collisions aux algorithmes de dynamique présentés dans le chapitre 5. La méthode de couplage est la même (recours à une distance de sécurité entre les objets, *cf* chapitre 6).

La base de données choisie est celle du moteur du lève-vitre fournie par Renault (décrite au chapitre 6). Une trajectoire complexe unique a été enregistrée, faisant intervenir des configurations variées (objet mobile proche ou éloigné des objets statiques) et des cohérences temporelles plus ou moins importantes (objet mobile plus ou moins rapide). Certaines étapes de cette trajectoire sont visibles dans la figure 3.5. Cette trajectoire unique a ensuite été rejouée pour chaque optimisation et pour leurs variantes en enregistrant les différentes statistiques de la détection de collisions. Selon les cas, ces statistiques peuvent être :

Les temps sont exprimés en secondes. Le temps T représente le temps total passé à rejouer les actions de l'utilisateur, mais n'inclut pas le temps passé à afficher les images. Plus précisément, T est le temps total passé dans la fonction `NextStep` de la classe `cScene` (*cf* annexe A). Notons enfin que tous les tests ont été effectués sur un Pentium PC 1GHz. Selon les optimisations, le temps total T varie entre une trentaine de secondes et une minute, reflétant ainsi la variété et la complexité des situations testées au cours de la trajectoire.

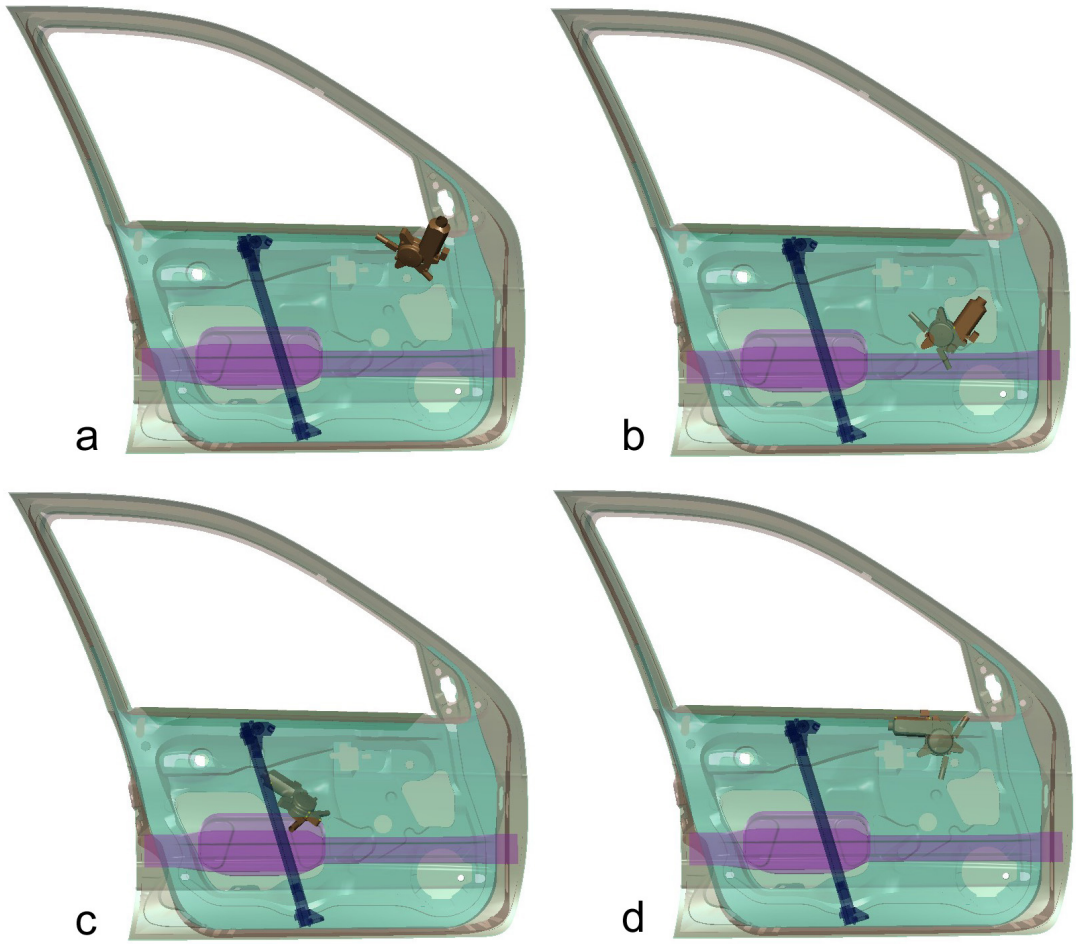


Fig. 3.5 – Extraits de la trajectoire de test utilisée. Les situations d'interaction typiques ont toutes été incluses. a : objet mobile éloigné de la portière. b : moteur en contact intérieur. c : près du rail du lève-vitre. d : mouvement rapide près de l'ajour.

3.5.1 Tests discrets

Afin de ne pas effectuer de trop nombreux tests OBB/OBB continus, un test discret de recouvrement est effectué aux positions initiales et finales avant chaque test continu. Si ces tests discrets montrent que les OBBs statiques se recouvrent à l'instant initial ou final, alors il n'est pas nécessaire d'effectuer le test recouvrement continu, puisque nous cherchons simplement à savoir si les volumes englobants se recouvrent au cours de l'intervalle de temps. Dans le cas contraire, c'est-à-dire lorsque les boîtes englobantes ne se recouvrent ni à l'instant initial, ni à l'instant final, il est bien sûr absolument nécessaire d'effectuer le test de recouvrement continu complet, afin de garantir que toutes les collisions seront détectées.

Puisque notre implantation du test continu est approximativement cinq fois plus lente que celle du test discret, ceci conduit à une accélération significative de la détection de collisions dans des situations à forte cohérence temporelle. Bien sûr, cette

N_{BV}	Nombre de tests BV/BV au cours de la simulation
N_{AA}	Nombre de tests A/A au cours de la simulation
N_{PF}	Nombre de tests P/F au cours de la simulation
N_{FP}	Nombre de tests F/P au cours de la simulation
N_{FP}	Nombre de tests F/P au cours de la simulation
T_{BV}	Temps total pour les tests BV/BV
T_{AA}	Temps total pour les tests A/A
T_{PF}	Temps total pour les tests P/F
T_{FP}	Temps total pour les tests F/P
T	Temps total pour la simulation

Tab. 3.1 – Notations utilisées lors des tests de la librairie.

optimisation est inutile dans des configurations générales.

Dans notre implantation, le test discret aux positions finales est exécuté, puis vient le test discret aux positions initiales, puis vient enfin le test continu. Il est bien entendu tout à fait possible d'intervertir les deux tests discrets, mais, d'une part, il est nécessaire de faire un choix et, d'autre part, ceci nous a semblé légèrement plus général, en ce sens qu'il permet davantage d'accélérer les situations ou un objet initialement éloigné de l'environnement vient en contact avec celui-ci. Bien sûr, il est possible de trouver des environnements pour lesquels les objets sont en permanence très proches du décor, ou très proches les uns des autres. Dans ce cas, il est sans doute préférable d'intervertir les deux tests discrets.

Notons qu'il est bien nécessaire d'effectuer les deux tests discrets. Le test discret final du pas de temps courant $[t_n, t_{n+1}]$ ne peut en effet pas servir de test discret initial pour le pas de temps suivant lorsqu'une collision a été détectée, puisque le pas de temps suivant commence alors *avant* t_{n+1}. Dans le cas où aucune collision n'a été détectée au pas de temps précédent, le test discret initial n'est bien sûr pas nécessaire.

Le tableau 3.2 récapitule les résultats des tests (dans le tableau, *test discret final* indique bien le test discret à l'instant final, et non pas le test discret réalisé en second). Il apparait que l'utilisation de tests discrets permet d'accélérer significativement la détection de collisions, et donc la simulation tout entière.

Remarquons que par sa nature même, cette optimisation permet seulement de réduire le nombre de tests de recouvrement continus entre volumes englobants, et n'a aucune influence sur le nombre de détection de collisions élémentaires effectuées. Ceci est d'ailleurs visible dans le tableau 3.2 : le gain de temps obtenu grâce aux tests discrets $(44,73 - 27,99 = 16,74s)$ se retrouve approximativement sur le gain de temps total pour la simulation T $(55,99 - 39,10 = 16,89s)$.

Notons aussi que T_{BV} représente le temps total requis par les fonctions de recouvrement entre volumes englobants, qu'elles soient continues ou discrètes. Il est possible de détailler les résultats du tableau 3.2 en recensant le nombre total de tests discrets

	T_{BV} (secondes)	T (secondes)
Sans tests discrets	44,73	55,99
Test discret final seul	30,75	41,83
Test discret initial seul	30,11	41,21
Tests discrets initial et final	27,99	39,10

Tab. 3.2 – L'utilisation de tests OBB/OBB discrets aux positions initiales et finales des objets avant le test de recouvrement continu permet d'accélérer significativement la détection de collisions.

initiaux N^i_{BV}, finaux N^f_{BV}, et continus N^c_{BV}, ainsi que les temps totaux respectifs T^i_{BV}, T^f_{BV} et T^c_{BV}. Il est alors possible de décomposer T_{BV} :

$$T_{BV} = T^i_{BV} + T^f_{BV} + T^c_{BV} = (N^i_{BV} + N^f_{BV}) \times C^d_{BV} + N^c_{BV} \times C^c_{BV} \tag{3.19}$$

où C^d_{BV} et C^c_{BV} sont respectivement le coût moyen d'un test BV/BV discret et le coût moyen d'un test BV/BV continu. Cette équation de coût montre clairement que l'accélération résultant de cette optimisation dépend fortement de la nature du mouvement de l'objet et de sa position par rapport à l'environnement. Ainsi, dans le cas d'une forte cohérence temporelle, comme c'est le cas par exemple lors du positionnement précis d'un objet, ou bien lors de manipulation d'un objet en contact, l'accélération de la détection de collisions peut être beaucoup plus importante que celle donnée dans le tableau 3.2, qui représente une trajectoire mixte. Plus la cohérence temporelle est importante, et plus cette optimisation permet d'éviter un nombre important de tests continus.

Notons cependant que, même dans le cas d'une forte cohérence temporelle (et même dans le cas limite d'un objet immobile), les volumes englobants qui ne pénètrent pas un autre volume englobant à l'instant initial ou final doivent impérativement faire l'objet d'un test continu. Pour tirer parti de la cohérence temporelle *en elle-même*, et éviter certains tests continus, il faudrait par exemple utiliser le mouvement des boîtes englobantes, à la manière de Mirtich [Mir96], pour estimer une borne inférieure sur l'instant de premier recouvrement entre boîtes englobantes. Ceci nous a cependant semblé *a priori* trop coûteux, en raison notamment de la nécessité de calculer avant l'estimation de la borne inférieure les points les plus proches appartenant aux deux boîtes englobantes.

Si l'on considère de nouveau la trajectoire utilisée dans le tableau 3.2, et que l'on décompose T_{BV}, on obtient les résultats suivants :

Ce tableau appelle deux remarques. Tout d'abord, il est possible de constater que, bien que les deux tests discrets utilisés séparément sont à peu près aussi efficaces l'un que l'autre, il est encore plus efficace de les utiliser ensemble, et ce malgré le surcoût engendré. La dernière ligne du tableau 3.3 montre en effet que, même après le test discret final, le test discret initial permet d'éliminer encore plus de 6% des tests continus.

	T^f_{BV}	N^f_{BV}	T^i_{BV}	N^i_{BV}	T^c_{BV}	N^c_{BV}
Sans tests discrets	-	-	-	-	44,73	5.050.305
Test discret final seul	7.37	5.050.305	-	-	23,39	2.577.968
Test discret initial seul	-	-	4,51	5.050.305	25,60	2.653.086
Tests discrets initial et final	7,48	5.075.137	0,74	2.600.063	19,77	2.430.508

Tab. 3.3 – Décomposition du temps total T_{BV} passé dans les fonctions de tests de recouvrement entre volumes englobants.

Une deuxième remarque est que le test discret final seul est plus coûteux qu'un test discret initial seul (la comparaison est pertinente puisque les deux tests utilisés seuls permettent d'éliminer approximativement le même nombre de tests continus). Ceci est dû au surcoût engendré par le repositionnement temporaire des boîtes englobantes à l'instant final.

Notons enfin que si l'on utilise les deux dernières ligne du tableau 3.3 pour évaluer le coût moyen d'un test discret initial, on trouve deux résultats fortement différents : $0,89\mu s$ lorsque le test est utilisé seul, et $0,28\mu s$ lorsque les deux tests discrets sont utilisés. Ceci est dû à deux raisons. La première est que les deux tests discrets sont *factorisables* : il est possible de réutiliser des calculs effectués dans le test discret final afin d'accélérer le test discret initial, ce dont nous avons tiré parti lors de l'implantation. La deuxième raison est que lorsque le test discret à l'instant final est utilisé, la plupart des tests discrets initiaux concluent à une séparation des boîtes englobantes, et peuvent être terminés avant de tester l'ensemble des quinze axes séparateurs.

3.5.2 Evaluation du test de subdivision

Nous avons cherché à évaluer l'importance du test de subdivision, afin en particulier d'évaluer la constante k qu'il fait intervenir (*cf* section 3.3.1). Afin de créer un environnement de test le plus réaliste possible, nous avons utilisé la même trajectoire que pour le test précédent, en conservant les tests discrets.

Les résultats sont rapportés dans la figure 3.6, qui montre l'évolution de T_{BV} et T (respectivement le temps total pris par les tests de recouvrement entre volumes englobants et le temps total pris pour rejouer la simulation) en fonction de la constante k. Ces évolutions s'expliquent simplement par la nature du test de subdivision. Rappelons que ce test est une heuristique dont le but est de déterminer les cas où les tests d'axes séparateurs continus annoncent à tort un recouvrement des OBBs durant un intervalle de temps donné. Plus la constante k est grande, et moins l'intervalle de temps a de chances d'être subdivisé. Deux effets contraires entrent alors en jeu. D'un côté, la réduction du nombre de subdivisions tend à diminuer le coût moyen d'*un* test de recouvrement. De l'autre côté, cependant, elle tend aussi à augmenter le nombre de tests de recouvrement.

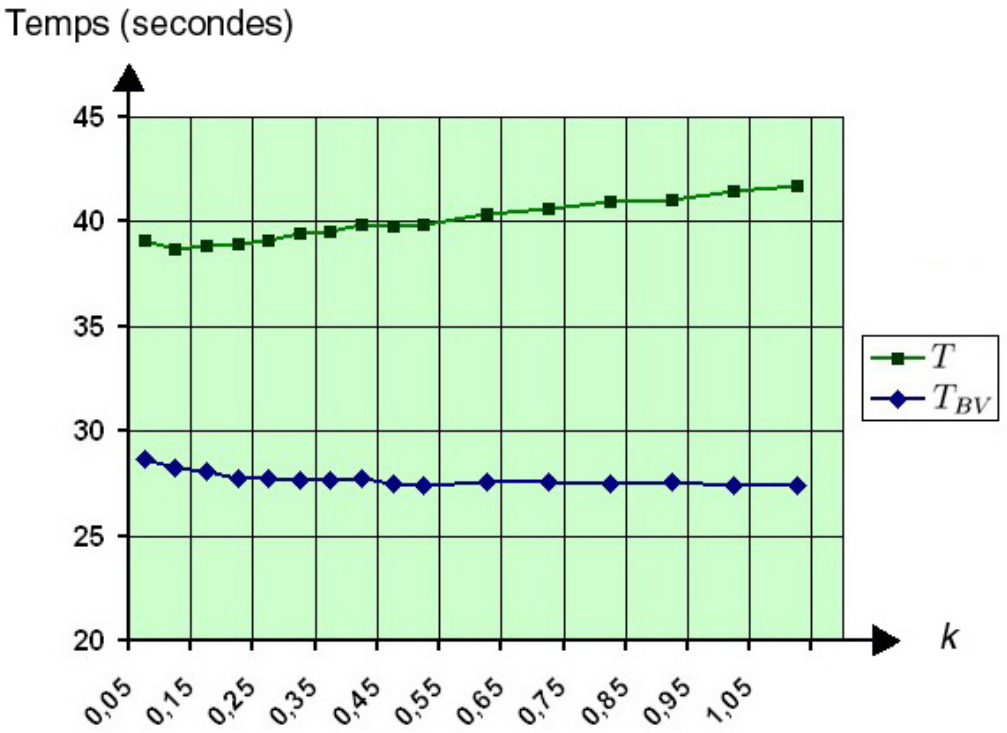


Fig. 3.6 – Evolution de T_{BV} et T en fonction de la constante k.

3.5.3 Passage du temps courant de collision

Dans la première approche, nous avons proposé de passer *l'instant de premier contact courant* t_c aux fonctions de détection de recouvrement entre volumes englobants. Cette optimisation est bien sûr encore valable et s'incrit naturellement dans l'utilisation de l'arithmétique d'intervalles. Bien que les boîtes orientées constituent une meilleure approximation des objets, cette optimisation permet encore d'accélérer significativement la simulation, comme le montre le tableau 3.4.

	N_{BV}	N_{AA}	N_{PF}	N_{FP}	T
Sans passage de t_c	6.685.889	7.226.139	2.335.435	2.363.427	79.59s
Avec passage de t_c	5.075.137	3.191.041	1.036.723	1.050.031	38.93s

Tab. 3.4 – Influence du passage de l'instant courant de première collision à la fonction de détection de recouvrement entre volumes englobants.

Il apparait que, comme dans le chapitre précédent, cette optimisation permet de réduire significativement le nombre de tests élémentaires, ainsi que le nombre de tests de recouvrement, et entraîne un diminution du temps total T d'un facteur deux.

3.5.4 Tests partiels

Van den Bergen [VDB98] remarque qu'un axe séparant deux OBBs est généralement un axe de l'un des OBBs, et supprime les neuf autres tests. Ceci conduit en fait à un ralentissement de la détection de collisions dans notre cas. Nous avons en effet observé que ces neuf axes sont souvent utiles pour séparer des objets en rotation au cours de l'intervalle de temps, et ainsi réduire le nombre de subdivisions de l'intervalle de temps.

Nous avons évalué l'apport des tests partiels dans le cas où les tests discrets initiaux et finaux sont utilisés conjointement avec le test continu. Quatre situations différentes ont été testées, selon que les tests complets étaient effectués pour les tests discrets ou continus. Le tableau 3.5 donne les temps T_{BV} et T dans chacune des quatre situations : Les neuf tests d'axes séparateurs supplémentaires, malgré leur coût propre,

	T_{BV}	N_{BV}	T
Discrets et Continu partiels	29,03	6.629.533	46,15
Discrets complet seul	214,61	6.602.937	231,65
Continu complet seul	26,93	6.499.411	43,56
Discrets et Continu complets	27,82	5.075.137	38,63

Tab. 3.5 – Evaluation de l'utilisation de tests partiels.

permettent de réduire le nombre de tests de recouvrement entre volumes englobants et donc d'accélérer la détection de collisions.

Notons que le cas le pire est celui pour lequel le test discret est complet alors que le test continu n'est que partiel. Ceci s'explique par les finalités différentes des tests discrets et continus. Les tests discrets, en effet, ne sont utiles *que lorsque les boîtes se recouvrent*, en permettant alors d'éviter des tests continus. Lorsque ces tests discrets sont complets, ils concluent plus souvent à une séparation des boîtes, et entraînent donc plus souvent des tests continus. Cependant, puisque ces tests continus sont partiels, leur première étape (la version continue des tests d'axe séparateurs) conclut moins souvent à une séparation des boîtes, et entraîne donc beaucoup plus souvent une subdivision de l'intervalle de temps en deuxième étape, augmentant ainsi le coût du test de recouvrement.

Notons en effet qu'en conservant cette situation (test discret complet et test continu partiel), et en supprimant l'étape de subdivision, on obtient des statistiques de simulation plus acceptables : $T_{BV} = 32,23$, $N_{BV} = 7.725.989$ et $T = 57,08$. Cependant, cette configuration reste moins efficace que celle où les tests discrets et continus sont complets, et où l'étape de subdivision est autorisée.

Comme précédemment, il est possible d'affiner l'analyse de cette optimisation en décomposant T_{BV} en fonction des statistiques des tests discrets et continus. Les résultats sont donnés dans le tableau 3.6.

	T^f_{BV}	N^f_{BV}	T^i_{BV}	N^i_{BV}	T^c_{BV}	N^c_{BV}
Discrets et Continu partiels	9,35	6.629.533	0,80	3.198.136	18,88	2.967062
Discrets complet seul	9,40	6.602.937	1,17	4.028.541	204,05	2.845.285
Continu complet seul	9,12	6.499.411	0,77	3.082.978	17,05	2.856.755
Discrets et Continu complets	7,41	5.075.137	0,76	2.600.063	19,71	2.430.508

Tab. 3.6 – Evaluation de l'utilisation de tests partiels. Décomposition de T_{BV}.

Le phénomène précédent apparaît encore un peu plus clairement dans la deuxième ligne du tableau 3.6 : lorsque les tests discrets sont complets, le test discret final (*i.e.* aux positions finales), exécuté en premier, conclut plus souvent à une séparation des boîtes, et conduit à une augmentation des tests discrets initiaux.

3.5.5 Tables de cohérence

Nous avons implanté la méthode des tables de cohérence introduite dans la thèse de Gottschalk [Got99]. En bref, la méthode consiste à stocker les noeuds terminaux de l'arbre des tests de volumes englobants. Ainsi, afin de détecter une collision entre deux objets, il suffit de commencer par les noeuds terminaux de la détection précédente. Cette méthode peut requérir une très grande quantité de mémoire, en particulier pour les objets complexes. De plus, à cause de l'utilisation des tests discrets préliminaires, cette optimisation ne conduit qu'à une faible accélération. Pour ces deux raisons, nous avons abandonné cette méthode.

3.5.6 Autres optimisations

De façon surprenante, une technique d'optimisation bien connue n'a pas permis d'accélérer la détection de collisions. Des valeurs précalculées des fonctions trigonométriques requises par le calcul des fonctions d'inclusions (*cf* équation (3.3)) étaient stockées dans des tables, sans toutefois donner un résultat significatif. Nous conjecturons que le compilateur (ou le processeur) utilise déjà une telle stratégie. Une autre explication possible est la lenteur de l'accès à la mémoire, comparée à la vitesse de l'évaluation des fonctions trigonométriques.

Certaines optimisations de bas niveau restent à essayer. Par exemple, de nombreux processeurs récents offrent des possibilités de parallélisme local au moyen d'instructions SIMD (*single instruction, multiple data* : une seule instruction, plusieurs données). Lin *et al.* rapportent qu'une version SIMD du test OBB/OBB discret permet de l'accélérer d'un facteur compris entre deux et trois [LGEGT99]. Notre test continu devrait bénéficier également d'une version SIMD. Comme le profilage des fonctions révèle qu'à peu près la moitié du temps total passé à détecter une collision est prise par les

tests de recouvrement entre OBBs, ceci pourrait accélérer l'ensemble de la simulation dynamique de façon significative.

3.6 Conclusion

Dans ce chapitre, nous avons présenté une deuxième approche permettant de détecter des collisions entre objets polyédriques quelconques de manière continue et efficace. Dans cette deuxième approche, nous avons relâché la contrainte que nous nous étions imposés dans le chapitre précédent, qui consistait à rechercher des mouvements intermédiaires arbitraires permettant de résoudre algébriquement les équations de détection de collisions. Nous avons vu en effet que cette contrainte nous avait conduit à deux problèmes. D'une part, il n'était pas possible en théorie de détecter des collisions entre plusieurs objets mobiles, en raison de l'impossibilité de composer les mouvements de vissage (du moins pour obtenir des solutions algébriques aux équations de détection de collisions). D'autre part, nous avions été incités à utiliser des sphères, afin d'obtenir des équations de détection de recouvrement à solutions algébriques. Les sphères étant de piètres volumes englobants lorsque les objets sont plats ou fins, nous avions dû développer une optimisation spécifique, la méthode des étiquettes, pour accélérer la détection de collisions.

A la place, nous avons cherché dans cette deuxième approche à utiliser des volumes englobants plus efficaces que les sphères. Cette démarche nous a amené à considérer les boîtes englobantes orientées en raison des très bonnes performances qu'elles offrent en détection de collisions discrète. Afin de dériver un test de recouvrement continu entre boîtes englobantes orientées du test bien connu de recouvrement discret, nous avons proposé d'utiliser l'arithmétique d'intervalles. Plus précisément, nous avons proposé un test de recouvrement continu en deux étapes : dans une première étape, l'arithmétique d'intervalles est utilisée pour effectuer une version continue des quinze tests d'axe séparateur et déterminer si l'un au moins de ces axes est séparateur durant l'intervalle de temps considére puis, dans une deuxième étape, une heuristique fondée sur la vitesse des boîtes englobantes est employée afin de déterminer si la première étape a conclu à tort que les boîtes se recouvrent et si l'intervalle de temps devrait être subdivisé.

Outre la possibilité de dériver un test de recouvrement continu entre boîtes englobantes, l'arithmétique d'intervalles nous permet de résoudre les équations de détections de collisions entre primitives polyédriques de façon robuste, tout en résolvant naturellement le problème de la précision de la détection de collisions. Puisque nous ne cherchons plus, dans cette deuxième approche, à résoudre algébriquement les équations de détection de collisions, nous pouvons maintenant prendre en compte le cas où plusieurs objets sont mobiles en même temps, et les chaînes poly-articulées acycliques : la figure 3.7 montre le cas de six cubes mobiles, dont l'un est manipulé par l'utilisateur et peut être utilisé pour déplacer les autres. Dans la partie droite de l'image, les cubes

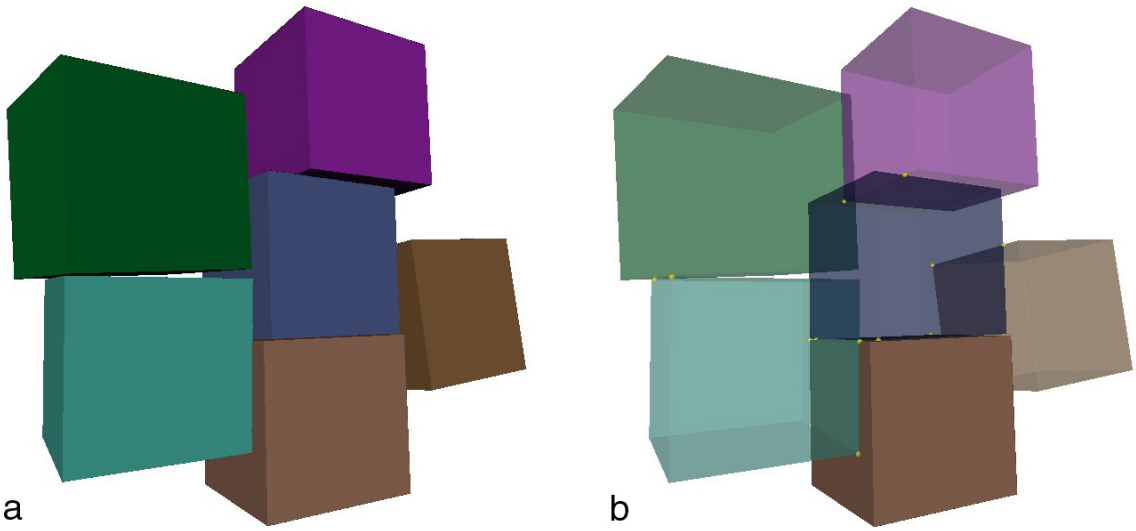


Fig. 3.7 – Six cubes mobiles. L'un des cubes est manipulé par l'utilisateur pour déplacer les autres. Dans la partie droite de l'image, les cubes sont affichés en transparence, à l'exception de celui manipulé par l'utilisateur, afin de mieux visualiser les points de contact (matérialisés par les sphères jaunes).

sont affichés en transparence, à l'exception de celui manipulé par l'utilisateur, afin de mieux visualiser les points de contact (matérialisés par les sphères jaunes).

Par ailleurs, nous avons remarqué que le test de recouvrement continu que nous proposons peut être utilisé pour accélérer les techniques antérieures de détection de collisions entre surfaces paramétrées ou implicites.

Enfin, nous avons étudié plusieurs optimisations et avons montré que deux d'entre elles permettent d'accélérer très significativement la détection de collisions : l'utilisation de tests discrets préalables d'une part, et le passage de l'instant de premier contact courant d'autre part. Notons que nous pouvons déjà déduire de ces tests que la méthode proposée dans ce chapitre, combinée aux optimisations, permet une interaction en temps réel sur un PC doté d'un processeur pentium 1GHz, et ce malgré la diversité et la complexité des configurations incluses dans la trajectoire de test. Sachant en effet que cette trajectoire comporte 2604 actions (et donc images), et que le temps total de la simulation est $T = 39,1s$ (*cf* dernière ligne du tableau 3.2), le taux de rafraîchissement moyen de la simulation est approximativement de 66 images par seconde.

Dans le chapitre suivant, nous allons décrire une autre méthode permettant d'accélérer la détection de collisions. Cette méthode, beaucoup plus générale puisqu'elle peut être appliquée à n'importe quelle méthode de détection de collisions fondée sur des hiérarchies de volumes englobants, qu'elle soit discrète ou continue, repose sur l'exploitation du mouvement des objets. Essentiellement, cette méthode d'accélération est motivée par le fait que les hiérarchies de volumes englobants perdent en efficacité lorsque les objets se rapprochent les uns des autres. Dans bien des cas cependant, et

plus encore lorsqu'un algorithme de calcul de mouvement contraint est utilisé et que les objets *glissent* les uns sur les autres, les parties des objets en contact ne se rapprochent pas les unes des autres, ou même *s'éloignent* les unes des autres. Aussi, nous allons maintenant décrire une méthode permettant de tirer parti des mouvements de recul des triangles associés aux volumes englobants, pour accélérer significativement la détection de collisions.

Chapitre 4

Tests de recul hiérarchiques et détection de collisions

Il y a quelques années, Vanecek [Van94] a proposé d'utiliser une variante d'une méthode d'élimination des parties cachées, fondée sur l'exploitation des normales aux faces, pour accélérer la détection de collisions entre objets polyédriques. Cependant, la méthode de Vanecek dépend linéairement du nombre de triangles composant les objets, ce qui la rend inutilisable pour des modèles complexes. Dans ce chapitre, nous proposons d'ajouter des informations géométriques aux hiérarchies de volumes englobants, typiquement utilisées dans les méthodes de détection de collisions, pour effectuer un test de recul hiérarchique au niveau du volume englobant, et ce en temps constant. La méthode que nous décrivons peut être utilisée pour compléter n'importe quel type de hiérarchie de volumes englobants, et autorise un compromis entre la mémoire requise et l'accélération résultante. Des résultats expérimentaux montrent que cette méthode permet d'obtenir une accélération significative, en particulier dans les situations où les objets sont très proches les uns des autres.

4.1 Introduction

Nous venons de montrer que la notion de mouvement intermédiaire arbitraire, alliée à la combinaison de l'arithmétique d'intervalles et des hiérarchies de boîtes englobantes, permet de détecter des collisions en continu assez efficacement pour interagir en temps réel avec des objets polyédriques (comme par exemple le modèle du moteur du lève-vitre, provenant d'un cas industriel).

Pourtant, bien que les hiérarchies de volumes englobants constituent un atout fondamental des méthodes de détection de collisions et soient essentielles dans notre approche, elles se révèlent de moins en moins efficaces à mesure que les objets se rapprochent les uns des autres, et que de plus en plus de volumes englobants se recouvrent. La cause principale de ce problème est que les volumes englobants ne permettent de

discriminer les objets *qu'en fonction de leur position*. Ce sont en effet leurs positions, et leurs positions seules, qui permettent de déterminer si les volumes englobants se recouvrent à un instant donné ou non. Que les tests de recouvrement soient effectués en continu au cours d'un intervalle de temps ne change pas la nature du problème : la notion de recouvrement est une notion spatiale, qui ne tient pas compte du mouvement des objets.

Dans ce chapitre, nous présentons une méthode permettant de tirer parti de la *vitesse* des objets et non plus seulement de leur position, comme le font les volumes englobants. Cette méthode est une généralisation d'une technique présentée par Vanecek en 1994 qui consiste à 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]. Puisqu'elle ne dépend que des vitesses relatives des triangles, et non de leur position ou orientation, la détection de collisions est particulièrement accélérée quand les objets sont proches les uns des autres, et qu'un grand nombre de paires de triangles doit être traité. Dans bien des cas en effet, en particulier lorsqu'un algorithme de calcul de mouvement contraint est utilisé et que les objets peuvent *glisser* les uns sur les autres, les parties proches des parties en contact ne se rapprochent pas les unes des autres, et parfois même *s'éloignent* les unes des autres. Aussi, dans d'autres cas, la nature même de l'interaction fait qu'une partie de l'objet manipulé est toujours en train de reculer. C'est le cas par exemple du classique *peg-in-a-hole*. Lorsque le bâton est dans l'orifice, à peu près la moitié des faces du bâton recule à chaque instant, relativement à l'orifice.

A notre connaissance, la méthode de Vanecek est la seule qui permet de discriminer des objets polyédriques quelconques, c'est-à-dire ne présentant pas de structure particulière, en fonction de leur vitesse seulement. Pour des objets convexes cependant, les algorithmes assimilés à la méthode du plan séparateur (*cf* Chung [Chu96]) sont à peu près insensibles aux positions et orientations des objets, et ce même lorsque les objets sont proches. Par ailleurs, pour des surfaces paramétriques ou implicites, la contrainte de rapprochement (*incoming constraint*) dans Snyder *et al.* [SWFCB93] peut être comprise comme une exploitation du mouvement de recul pour la détection de collisions. Remarquons toutefois que, même si certaines méthodes de détection de collisions utilisent le mouvement des objets pour borner inférieurement les instants de premiers contacts entre objets, comme celle de Mirtich [Mir96], elles n'utilisent pas la *direction* du mouvement, comme le fait Vanecek.

En quelques mots, l'idée de Vanecek consiste à effectuer un test de recul *élémentaire*, au niveau de la face, en comparant la normale à la face aux vitesses relatives des sommets de la face. Si la normale est dirigée vers l'extérieur de l'objet, et que les vitesses relatives sont dirigées vers l'intérieur de l'objet, alors la face recule relativement à l'autre objet, et ne peut entrer en contact avec lui.

Malheureusement, cette méthode dépend linéairement du nombre de faces de l'objet, ce qui la rend rapidement impraticable lorsque la complexité des objets augmente. Une combinaison naturelle des hiérarchies de volumes englobants et de la méthode de

Vanecek est bien sûr envisageable. Il suffirait en effet de n'effectuer les tests de recul que lorsque des volumes englobants se recouvrent. Cependant, cette méthode nécessiterait de descendre jusqu'aux feuilles des hiérarchies de volumes englobants avant d'exploiter le recul local de l'objet. Sans descendre jusqu'au feuilles, elle ne permettrait pas de détecter les situations où tous les triangles *associés* à un volume englobant, c'est-à-dire tous les triangles contenus dans les feuilles de sa descendance, reculent en même temps. Dans de telles situations pourtant, il n'est pas nécessaire de parcourir la descendance du volume englobant, et ce *même lorsqu'il recouvre un volume englobant d'un autre objet*.

Le but de ce chapitre est de développer un test de recul *hiérarchique*, au niveau du volume englobant, permettant de détecter ces situations où tous les triangles associés reculent, sans parcourir la descendance du volume englobant. Afin de réaliser ces tests de recul hiérarchiques, nous proposons d'ajouter des informations géométriques 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.

La suite de ce chapitre est essentiellement une formalisation de cette idée. Dans la section 4.2, nous rappelons la méthode de Vanecek permettant de faire des tests de recul au niveau du triangle. Cette méthode est ensuite généralisée dans la section 4.3 en introduisant les cône de reculs ajoutés aux volumes englobants. Les deux sections suivantes détaillent la façon d'ajouter et de construire les cônes de recul. La section 4.4 explique l'intégration des tests de recul hiérarchiques dans l'algorithme récursif de parcours des hiérarchies de volumes englobants, et présente les différentes façons d'ajouter les cônes de recul, tandis que la section 4.5 propose un algorithme de construction des cônes de recul eux-mêmes. La section 4.6 montre comment les tests de recul hiérarchiques s'intègre dans une approche de détection de collisions continue, et en particulier dans les algorithmes décrits au chapitre précédent. L'apport des tests de recul hiérarchiques est évalué dans la section 4.7. Enfin, la section 4.8 conclut le chapitre.

Notons que la méthode décrite dans ce chapitre permet de compléter tout type de volume englobant, et ne requiert pas de hiérarchie séparée. Nous verrons qu'elle autorise un compromis entre le surcoût de mémoire, entraîné par l'ajout des cônes de recul dans les volumes englobants, et l'accélération résultante.

4.2 Tests de recul élémentaires

Commençons par rappeler brièvement les résultats de Vanecek [Van94]. Nous renvoyons le lecteur à l'article initial pour plus de détails (nous employons les mêmes notations).

Nous avons vu que l'idée de Vanecek consiste à éliminer les faces qui reculent, c'est-à-dire celles dont les vitesses relatives aux sommets ne sont pas orientées dans le même sens que leur normale extérieure.

Cette idée se formalise simplement. Considérons un point $\mathbf{p}$ d'un objet rigide. Notons $\mathbf{r}$ le centre de gravité de l'objet, et notons respectivement $\dot{\mathbf{r}}$ et ω sa vitesse en translation et sa vitesse en rotation, à un instant donné. Alors la vitesse instantanée du point $\mathbf{p}$ de l'objet est :

$$\dot{\mathbf{p}} = \dot{\mathbf{r}} + \omega \wedge (\mathbf{p} - \mathbf{r}) \tag{4.1}$$

Considérons maintenant deux objets i et j. La vitesse relative du point $\mathbf{p}$ de l'objet i, par rapport à l'objet j, est :

$$\dot{\mathbf{p}}_{ij} = \dot{\mathbf{p}}_i - \dot{\mathbf{p}}_j = \mathbf{a}_{ij} + \mathbf{p} \wedge (\omega_j - \omega_i) \tag{4.2}$$

où $\mathbf{a}_{ij} = \dot{\mathbf{r}}_i - \dot{\mathbf{r}}_j - \omega_i \wedge \mathbf{r}_i + \omega_j \wedge \mathbf{r}_j$.

Il est important de constater que les vitesses relatives des points d'un objet sont linéairement reliées entre elles. Plus précisément, si $\mathbf{p}^1, \ldots, \mathbf{p}^t$ sont des points de l'objet i, alors

$$\mathbf{p} = \sum_{s=1}^{t} \alpha_s \mathbf{p}^s \Rightarrow \dot{\mathbf{p}}_{ij} = \sum_{s=1}^{t} \alpha_s \dot{\mathbf{p}}^s_{ij} \tag{4.3}$$

pour tout $(\alpha_1, ..., \alpha_t)$ dans $\mathbb{R}^t$.

Si l'on suppose maintenant que le point $\mathbf{p}$ appartient à une face m de l'objet i, et que $\mathbf{n}_m$ est la normale extérieure à cette face m, alors $\mathbf{p}$ recule relativement à l'objet j si et seulement si

$$\dot{\mathbf{p}}_{ij}.\mathbf{n}_m < 0 \tag{4.4}$$

La relation (4.3) et la linéarité du produit scalaire conduisent à la *propriété de convexité* suivante :

$$(\dot{\mathbf{p}}^s_{ij}.\mathbf{n}_m < 0, \quad 1 \leqslant s \leqslant t) \Rightarrow (\dot{\mathbf{p}}_{ij}.\mathbf{n}_m < 0) \tag{4.5}$$

pour tout point $\mathbf{p}$ dans l'enveloppe convexe de $\mathbf{p}^1, \ldots, \mathbf{p}^t$.

Cette propriété est au coeur de l'algorithme de Vanecek, qui élimine les faces en deux étapes :

1. Calcul des vitesses relatives des points extrémaux
Un polygone convexe englobant est associé à chaque face (si la face est un triangle, ce polygone englobant est le triangle lui-même). Les vitesses relatives des sommets $\mathbf{p}^1, \ldots, \mathbf{p}^t$ du polygone englobant (*i.e.* les points *extrémaux*) sont calculées grâce à l'équation (4.2).

2. Tests de recul des points extrémaux
Pour chaque sommet $\mathbf{p}^s$ du polygone englobant, un test de recul est effectué comme dans l'équation (4.4). Si tous les sommets $\mathbf{p}^s$ reculent, alors la face est éliminée.

L'utilisation d'un polygone convexe englobant permet essentiellement d'accélérer les tests pour les faces non convexes ou trop complexes. Par exemple, une face comportant trop de sommets est englobée dans un polygone comportant un nombre de sommets réduit. La propriété de convexité (4.5) assure alors que la face recule lorsque le polygone qui l'englobe recule.

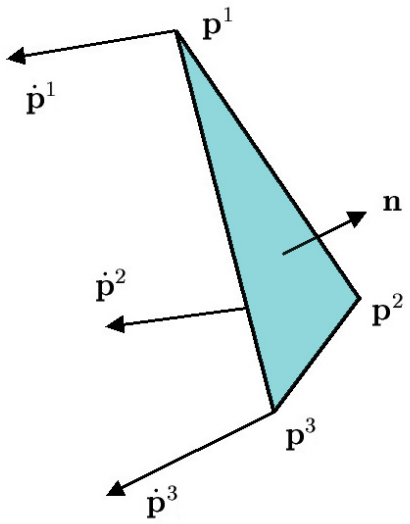


Fig. 4.1 – Détermination du recul d'un triangle. Le triangle recule relativement à l'autre objet lorsque les vitesses relatives de ses trois sommets sont dans le sens opposé à celui de la normale au triangle dirigée vers l'extérieur de l'objet.

La figure 4.1 montre un triangle de sommets $\mathbf{p}^1$, $\mathbf{p}^2$ et $\mathbf{p}^3$, et de normale $\mathbf{n}$. Les vitesses des sommets, relativement à un autre objet, sont $\dot{\mathbf{p}}^1$, $\dot{\mathbf{p}}^2$ et $\dot{\mathbf{p}}^3$. Notons que ces vitesses ne sont pas nécessairement parallèles lorsque la vitesse rotationnelle de l'objet est non nulle (*cf* équation (4.1)). Les trois produits scalaires $\dot{\mathbf{p}}^1.\mathbf{n}$, $\dot{\mathbf{p}}^2.\mathbf{n}$ et $\dot{\mathbf{p}}^3.\mathbf{n}$ étant négatifs, le triangle recule relativement à l'autre objet.

L'algorithme de Vanecek appelle trois remarques :

- le recul relatif d'un triangle est une propriété intrinsèque. Une fois calculée la vitesse relative du triangle qui, elle, dépend bien évidemment de la vitesse de l'autre objet, le test de recul peut être effectué indépendamment des tests de recul des autres triangles composant l'objet, et même de ceux des triangles composant l'autre objet. Ce caractère intrinsèque se retrouvera dans les tests de recul hiérarchiques, et sera exploité lors de l'intégration des tests dans l'algorithme de parcours des hiérarchies englobantes, dans la section 4.4.
- le test proposé par Vanecek est seulement *conservatif* lorsqu'une face n'est pas convexe. En effet, une face non convexe peut reculer sans que son enveloppe convexe recule (la propriété de convexité (4.5) est nécessaire mais n'est évidemment pas suffisante). Dans ce cas, le test de Vanecek ne permet pas d'éliminer la face. Nous verrons que les tests de recul hiérarchiques proposés sont eux-aussi conservatifs, et ce même lorsque toutes les faces sont des triangles.

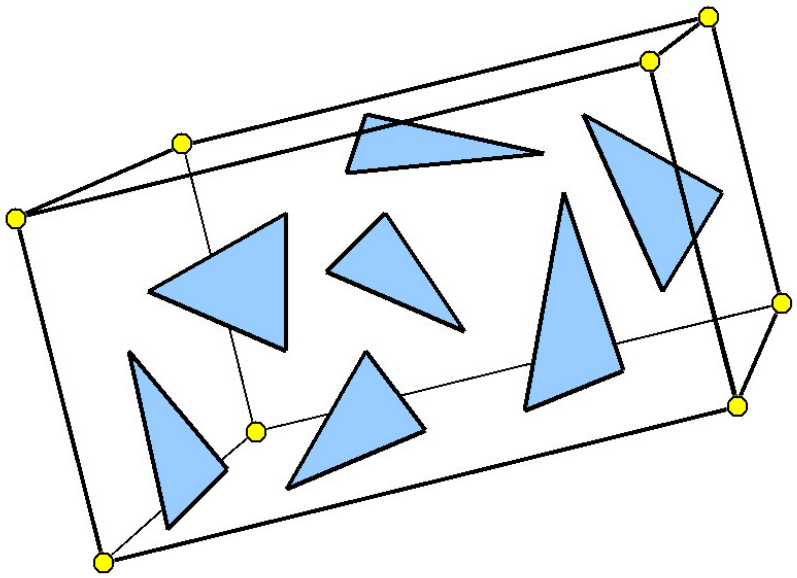


Fig. 4.2 – Les huit points caractéristiques associés à la boîte englobante orientée sont les sommets de la boîte.

– les calculs de vitesses relatives et les tests de recul (4.4) peuvent être intercalés afin de terminer le test d'une face plus vite lorsque l'un des sommets du polygone englobant ne recule pas. Bien sûr, intercaler les tests ne permet pas de réduire le nombre de triangles à tester.

4.3 Tests de recul hiérarchiques

Nous pouvons maintenant généraliser l'approche de Vanecek, essentiellement en approchant la géométrie des triangles associés au volume englobant, pour estimer leurs vitesses, et en approchant leurs normales. Pour cela, il faut adapter les deux étapes de l'algorithme de Vanecek. Quel que soit le nombre k de triangles associés au volume englobant, l'algorithme devient :

1. Calcul des vitesses relatives des points caractéristiques

Au lieu de calculer les vitesses relatives des $3k$ sommets définissant les k triangles associés, nous calculons les vitesses relatives d'un nombre fixe de *points caractéristiques* $\mathbf{c}^1, \ldots, \mathbf{c}^t$ dont l'enveloppe convexe contient le volume englobant. Pour une AABB ou une OBB, par exemple, les coins peuvent être utilisés ($t = 8$). Pour une sphère, les points caractéristiques doivent être situés à l'extérieur de la sphère. D'après la linéarité des vitesses relatives (propriété (4.3)), les vitesses des points caractéristiques sont une approximation des vitesses des triangles, puisque les triangles associés au volume englobant y sont contenus. Puisque le nombre de points caractéristiques ne dépend que du *type* du volume englobant (AABB, OBB...), cette première étape s'effectue en temps constant, et ce quel que soit le

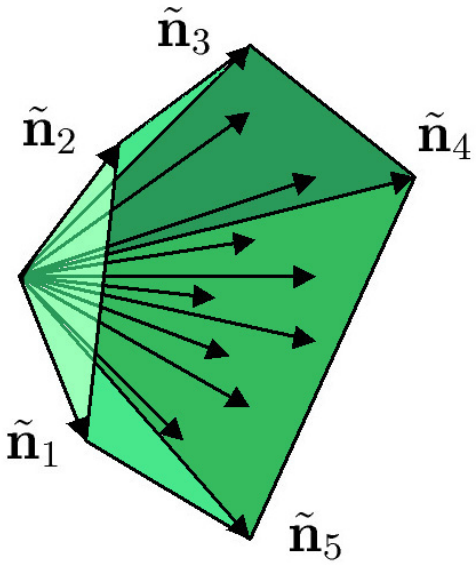


Fig. 4.3 – Les vecteurs de recul $\tilde{\mathbf{n}}_1$,$\tilde{\mathbf{n}}_2$,$\tilde{\mathbf{n}}_3$, $\tilde{\mathbf{n}}_4$ et $\tilde{\mathbf{n}}_5$ sont une approximation des normales aux triangles contenus dans le volume englobant.

nombre de triangles k qui lui sont associés. Les points caractéristiques associés à une boîte englobante orientée sont visibles en figure 4.2.

2. Tests de recul des points caractéristiques

D'après la propriété de convexité (4.5), les vitesses relatives des points caractéristiques sont une approximation des vitesses des sommets des triangles associés au volume englobant, et peuvent être utilisées pour éliminer les triangles qui reculent (plus précisément, les vitesses relatives des sommets des triangles sont des combinaisons linéaires positives de celles des points caractéristiques). Cependant, faire les tests pour chaque triangle serait encore en $O(k)$. Aussi utilisons-nous un nombre borné de *vecteurs de recul* précalculés $\tilde{\mathbf{n}}_1, \ldots, \tilde{\mathbf{n}}_r$ pour approcher l'ensemble des normales aux triangles (nous préciserons un peu plus loin la façon dont l'approximation doit être réalisée). Pour un point caractéristique $\mathbf{c}^s$ donné, on effectue avec chaque vecteur de recul précalculé $\tilde{\mathbf{n}}_m$ un test de recul comme dans l'équation (4.4). Le volume englobant est éliminé si chacun des t points caractéristiques reculent par rapport à chacune des r directions définies par les les vecteurs de recul. Un exemple de vecteurs de recul approchant l'ensemble des normales aux triangles est représenté dans la figure 4.3.

Afin de clarifier la suite du raisonnement et de simplifier les notations, il est pratique d'utiliser une représentation duale, plus visuelle.

Remarquons tout d'abord que le test de culling élémentaire (4.4) peut être interprété géométriquement. Notons

$$\mathbf{H}(\mathbf{n}_m) = \{\mathbf{x} \in \mathbb{R}^3,\ \mathbf{x}.\mathbf{n}_m < 0\} \tag{4.6}$$

le demi-espace vectoriel ouvert défini par $\mathbf{n}_m$. Avec cette notation, le point $\mathbf{p}$ de l'objet i recule relativement à l'objet j si et seulement si

$$\dot{\mathbf{p}}_{ij} \in \mathbf{H}(\mathbf{n}_m) \tag{4.7}$$

L'interprétation géométrique de cette relation est visible dans la figure 4.4.

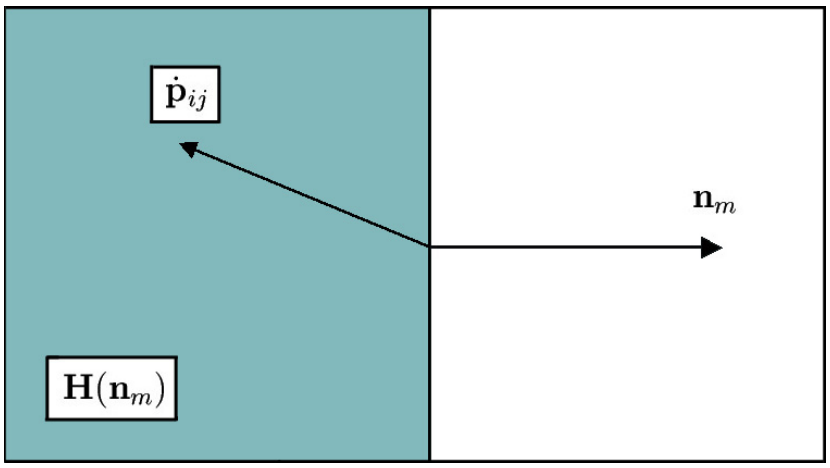


Fig. 4.4 – Le point $\mathbf{p}$ de l'objet i recule relativement à l'objet j si et seulement si sa vitesse relative $\dot{\mathbf{p}}_{ij}$ est dans le demi-espace ouvert $\mathbf{H}(\mathbf{n}_m)$.

Les vecteurs de recul précalculés doivent être choisis de façon à fournir un test de recul hiérarchique *conservatif* : il faut que lorsque le test de recul affirme que l'on peut éliminer le volume englobant, tous les triangles qui lui sont associés reculent effectivement. Cette contrainte peut également s'exprimer simplement : il suffit de choisir les vecteurs $\tilde{\mathbf{n}}_1, \ldots, \tilde{\mathbf{n}}_r$ de façon que

$$\mathbf{CR}_v := \bigcap_{m=1}^{r} \mathbf{H}(\tilde{\mathbf{n}}_m) \subset \bigcap_{m=1}^{k} \mathbf{H}(\mathbf{n}_m) \tag{4.8}$$

où $\mathbf{CR}_v$ est le cône de recul polyédrique, défini par les r vecteurs de recul, et ajouté au volume englobant v. L'entier r, la *taille* du cône de recul, est inférieur à une constante r_{max} définie par l'utilisateur.

En éliminant le volume englobant v quand

$$\dot{\mathbf{c}}_{ij}^{s} \in \mathbf{CR}_v, \ 1 \leqslant s \leqslant t \tag{4.9}$$

nous sommes assurés que le test de recul hiérarchique est conservatif, comme le montre le lemme suivant :

Lemme 1 (Conservativité des tests de recul hiérarchiques). *Si les vecteurs $\tilde{\mathbf{n}}_1, \ldots, \tilde{\mathbf{n}}_r$ vérifient la propriété d'inclusion (4.8), tout test de recul hiérarchique est conservatif. Autrement dit, tous les triangles associés au volume englobant reculent nécessairement lorsque le volume englobant est éliminé.*

Démonstration. Considérons un point $\mathbf{p}$, sommet de l'un des k triangles associés. Ce sommet est dans l'enveloppe convexe des points caractéristiques $\mathbf{c}^1, \ldots, \mathbf{c}^t$ associés au volume englobant.

Supposons maintenant que le test de recul hiérarchique (4.9) élimine le volume englobant, autrement dit que les vitesses relatives des points caractéristiques soient toutes dans le cône de recul. D'après la relation d'inclusion (4.8), ces vitesses relatives sont aussi dans $\bigcap_{m=1}^{k} \mathbf{H}(\mathbf{n}_m)$. Or, par linéarité des vitesses relatives, la vitesse relative du point $\mathbf{p}$ est dans l'enveloppe convexe des vitesses relatives des points caractéristiques. Comme l'ensemble $\bigcap_{m=1}^{k} \mathbf{H}(\mathbf{n}_m)$ est lui-même convexe, il contient la vitesse relative du point $\mathbf{p}$. Autrement dit, le point $\mathbf{p}$ recule, puisque la normale à la face dont $\mathbf{p}$ fait partie se trouve parmi les m normales $\mathbf{n}_1, \ldots, \mathbf{n}_m$ définissant $\bigcap_{m=1}^{k} \mathbf{H}(\mathbf{n}_m)$. □

Notons que, puisque t et r_{max} sont fixés, le test de recul hiérarchique est bien effectué en temps constant. Ici aussi, d'ailleurs, les calculs de vitesses et les tests de recul des points caractéristiques peuvent être intercalés, afin éventuellement de terminer le test de recul hiérarchique plus tôt lorsqu'un point caractéristique ne recule pas, sans avoir à effectuer l'ensemble des $t \times r$ tests.

Remarquons cependant que ce test n'est bien *que* conservatif : il peut déclarer *à tort* qu'un volume englobant ne recule pas, même si chaque triangle associé recule. Les raisons devraient maintenant en être claires : d'une part, les vitesses des triangles sont approchées par celles de quelques points caractéristiques et, d'autre part, les normales aux triangles sont approchées par le cône de recul. Cependant, les volumes englobants constituent une approximation de plus en plus précise de la géométrie de l'objet à mesure que l'on descend dans la hiérarchie, et les cônes de recul permettent d'éliminer une proportion de plus en plus importante des triangles qui reculent.

La figure 4.5 montre ce phénomène. Dans cet exemple, la portière s'éloigne de la caméra et presque toutes les faces reculent. Les images a, b, c et d représentent les résultats des tests de recul hiérarchiques pour les niveaux 7, 9, 11 et 14 de la hiérarchie englobant la portière, au même instant. Dans chacun de ces quatre cas, tous les volumes englobants du niveau correspondant ont subi un test de recul hiérarchique. Lorsque le test a conclu que les triangles associés reculent, ces triangles ont été affichés en transparence. Il apparaît clairement que les cônes de recul permettent d'éliminer de plus en plus de volumes englobants, et donc de plus en plus de groupes de triangles associés, à mesure que l'on descend dans la hiérarchie englobante. Il apparaît aussi que les cônes de recul permettent bien de ne pas avoir à descendre jusqu'aux feuilles de la hiérarchie englobante pour exploiter le recul de l'objet et accélérer la détection de collisions, en n'ayant pas à parcourir la descendance d'un volume englobant éliminé.

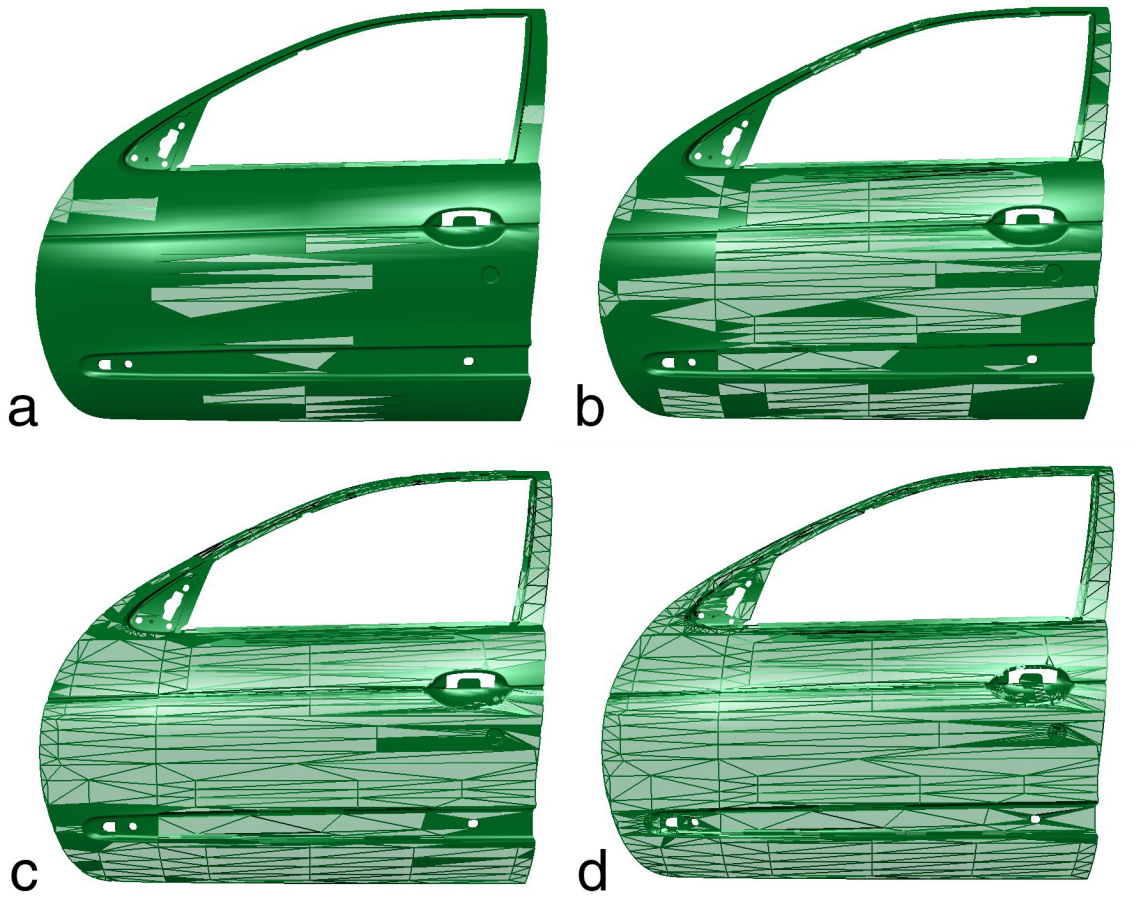


Fig. 4.5 – Dans cet exemple, la portière s'éloigne de la caméra et presque toutes les faces reculent. Les images a, b, c et d représentent les résultats des tests de recul hiérarchiques pour les niveaux 7, 9, 11 et 14 de la hiérarchie englobant la portière, au même instant. Dans chacun de ces quatre cas, tous les volumes englobants du niveau correspondant ont subi un test de recul hiérarchique. Lorsque le test a conclu que les triangles associés reculent, ces triangles ont été affichés en transparence. Les cônes de recul permettent d'éliminer de plus en plus de volumes englobants, et donc de plus en plus de groupes de triangles associés, à mesure que l'on descend dans la hiérarchie englobante.

4.4 Extension des hiérarchies de volumes englobants

4.4.1 Intégration des tests de recul hiérarchiques

Pour exploiter les cônes de recul, il suffit de modifier l'algorithme récursif de parcours des hiérarchies englobantes donné dans le chapitre 1. Plus précisément, le test de recouvrement de deux volumes englobants, première étape de l'algorithme, est remplacé par les étapes suivantes :

1. Si le noeud A a déjà été éliminé par un test de recul hiérarchique ($s^e_A = cdstamp$), FIN.
2. Si le noeud B a déjà été éliminé par un test de recul hiérarchique ($s^e_B = cdstamp$), FIN.
3. Si les noeuds A et B ne se recouvrent pas, FIN.
4. Si le noeud A n'a pas encore subi le test de recul ($s^t_A \neq cdstamp$) :
   (a) Marquer A ($s^t_A = cdstamp$).
   (b) Faire passer le test de recul hiérarchique à A.
   (c) Si A recule, le marquer ($s^e_A = cdstamp$), FIN.
5. Si le noeud B n'a pas encore subi le test de recul ($s^t_B \neq cdstamp$) :
   (a) Marquer B ($s^t_B = cdstamp$).
   (b) Faire passer le test de recul hiérarchique à B.
   (c) Si B recule, le marquer ($s^e_B = cdstamp$), FIN.

Rappelons, comme nous l'avons vu au chapitre 2, que l'entier `cdstamp` est incrémenté à chaque fois que l'algorithme récursif de parcours des hiérarchies englobantes traite une nouvelle paire d'objets. Ici, l'entier s^t_A (respectivement s^t_B) représente le moment ou le volume englobant A (respectivement B) à subi un test de volume hiérarchique pour la dernière fois. De même, l'entier s^e_A (respectivement s^e_B) représente le moment ou le volume englobant A (respectivement B) à été éliminé par un test de volume hiérarchique pour la dernière fois. Bien qu'un volume englobant puisse intervenir dans plusieurs tests de recouvrement, pour une même paire d'objets, ces entiers permettent d'éviter de lui faire passer plusieurs fois le test de recul hiérarchique. En effet, de même que le recul relatif d'un triangle est une propriété intrinsèque, qui ne dépend pas du recul des autres triangles de l'objet, ou même de ceux de l'autre objet (ainsi que nous l'avons vu dans la section 4.2), le recul relatif de l'ensemble des triangles associés à un volume englobant est lui aussi une propriété intrinsèque.

4.4.2 Construction des hiérarchies englobantes

Afin de tirer le meilleur parti des tests de recul hiérarchiques, une idée naturelle consiste à trier les triangles en fonction de leurs normales, lors de la construction des hiérarchies englobantes.

Ainsi, si l'on note $\mathbf{n}_1$, ... $\mathbf{n}_k$ les normales aux triangles associés au volume englobant courant, la moyenne des normales est

$$\mu = \frac{1}{k} \sum_{i=1}^{k} \mathbf{n}_i \tag{4.10}$$

et les composantes de leur matrice de covariance sont

$$\mathbf{C}_{lm} = \frac{1}{k} \sum_{i=1}^{k} (\mathbf{n}_i[l] - \mu[l])(\mathbf{n}_i[m] - \mu[m]) \tag{4.11}$$

où $1 \leqslant l, m \leqslant 3$, $\mathbf{n}_i[l]$ et $\mu[l]$ sont les coordonnées des vecteurs.

Cette matrice de covariance est ensuite diagonalisée et les triangles sont répartis en deux sous-liste selon la projection de leur normale le long de la direction de plus grande variance. Cette idée est semblable à celle utilisée pour construire des boîtes orientées englobant un ensemble de triangles.

Cependant, cette méthode fait l'impasse sur un point fondamental de la détection de collisions. En ne tenant pas compte de la *position* des triangles, elle a tendance à produire des volumes englobants trop grands.

Bien qu'une règle semblable à celle proposée par Kumar *et al.* [KMGL99], répartissant les triangles en fonction de leurs positions et orientations, pourrait être conçue, il apparaît en fait qu'une règle de répartition traditionnelle fondée uniquement sur la position des triangles, la règle *min-max* mentionnée au chapitre précédent par exemple, est suffisante. En effet, pour des objets réguliers du type de ceux que l'on rencontre par exemple dans l'industrie (surfaces au moins $\mathcal{C}^1$ par morceaux), la courbure locale est de plus en plus faible à mesure que l'on descend dans la hiérarchie englobante, et les normales varient de moins en moins, rendant le test de recul hiérarchique de plus en plus efficace, comme le montrent la figure 4.5 et l'évaluation de la section 4.7.

4.4.3 Ajout des cônes de recul

Les cônes de recul peuvent être ajoutés à chaque volume englobant indépendamment du type de méthode utilisée pour construire les hiérarchies. En particulier, il est possible d'opter pour une stratégie *top-down* ou *bottom-up*.

Dans le cas *top-down*, les cônes de recul sont calculés directement à partir des normales $\mathbf{n}_1, \ldots, \mathbf{n}_k$ aux triangles associés au volume englobant, alors que dans le cas *bottom-up*, les cônes de recul sont calculés récursivement à partir des cônes de recul des volumes englobants fils, selon les règles suivantes :

1. Le cône de recul d'une feuille de la hiérarchie est calculé à partir de la (des) normale(s) au(x) triangle(s) associé(s) à cette feuille.
2. Le cône de recul d'un noeud interne est calculé à partir des *vecteurs de recul* contenus dans les fils de ce noeud, à la condition que chaque noeud fils ait au moins de vecteur de recul[1].

Il est facile de se convaincre que la stratégie *bottom-up* produit bien des tests de recul hiérarchiques conservatifs : les volumes englobants ne peuvent pas être éliminés à tort. Si $\mathbf{CR}_{v_1}, \ldots, \mathbf{CR}_{v_c}$ sont les cônes de recul associés aux noeuds fils du volume englobant, alors

$$\mathbf{CR}_v \subset \bigcap_{i=1}^{c} \mathbf{CR}_{v_i} \tag{4.12}$$

[1] L'absence de vecteurs de recul signifiant en effet que le cône de recul est vide.

par associativité de l'intersection dans la relation (4.8).

Nous présentons à la section 4.7 les résultats de tests visant à comparer l'efficacité des deux stratégies, en termes de coûts de construction, d'encombrement mémoire et d'accélération de la détection de collisions.

4.5 Construction du cône de recul

Il est temps de décrire un algorithme permettant de construire le cône de recul d'un volume englobant. D'après la section précédente, le cône de recul est construit à partir de vecteurs qui sont soit les normales aux triangles associés, soit tous les vecteurs de recul des volumes englobants fils, lorsque tous ces volumes englobants fils ont un cône de recul.

4.5.1 Détermination des vecteurs de recul appropriés

Nous allons tout d'abord préciser la façon de choisir les vecteurs composant les cônes de recul. Pour cela, les définitions suivantes seront utiles :

Définition (Cône positif). *Le cône positif* $\mathcal{P}(\mathbf{v}_1, \ldots, \mathbf{v}_k)$ *engendré par* k *vecteurs* $\mathbf{v}_1$, $\ldots$, $\mathbf{v}_k$ *de* $\mathbb{R}^3$ *est l'ensemble des combinaisons linéaires positives de ces* k *vecteurs :*

$$\mathcal{P}(\mathbf{v}_1, \ldots, \mathbf{v}_k) = \left\{ \sum_{m=1}^{k} \lambda_m \mathbf{v}_m, \;\; \lambda_m \in \mathbb{R}^+, \;\; 1 \leqslant m \leqslant k \right\}$$

Définition (Sous-ensemble minimal). *Soient* $\mathbf{v}_1$, $\ldots$, $\mathbf{v}_k$, k *vecteurs de* $\mathbb{R}^3$. *Dans ce chapitre, un sous-ensemble minimal de ces vecteurs est un plus petit sous-ensemble de* t *vecteurs* $\mathbf{v}_{i_1}$, $\ldots$, $\mathbf{v}_{i_t}$ *de ces* k *vecteurs tel que*

$$\mathcal{P}(\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_t}) = \mathcal{P}(\mathbf{v}_1, \ldots, \mathbf{v}_k)$$

En d'autres termes, un sous-ensemble minimal est un plus petit sous-ensemble de vecteurs qui engendrent *le même cône positif*. Il est clair que les vecteurs d'un sous-ensemble minimal sont sur l'enveloppe convexe du cône positif initial. Il est clair aussi qu'il n'y a pas unicité du sous-ensemble minimal.

La figure 4.6 montre, dans le plan, le cône positif engendré par trois vecteurs $\mathbf{v}_1$, $\mathbf{v}_2$ et $\mathbf{v}_3$ du plan. Le vecteur $\mathbf{v}_2$ est redondant et le sous-ensemble minimal (unique dans ce cas), est composé des vecteurs $\mathbf{v}_1$ et $\mathbf{v}_3$. Il est clair que $\mathcal{P}(\mathbf{v}_1, \mathbf{v}_3) = \mathcal{P}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$.

La contrainte imposée sur le cône de recul est la propriété d'inclusion (4.8). Aussi, l'algorithme de construction utilise le lemme suivant pour déterminer des vecteurs $\tilde{\mathbf{n}}_1$, $\ldots$, $\tilde{\mathbf{n}}_r$ satisfaisant cette condition :

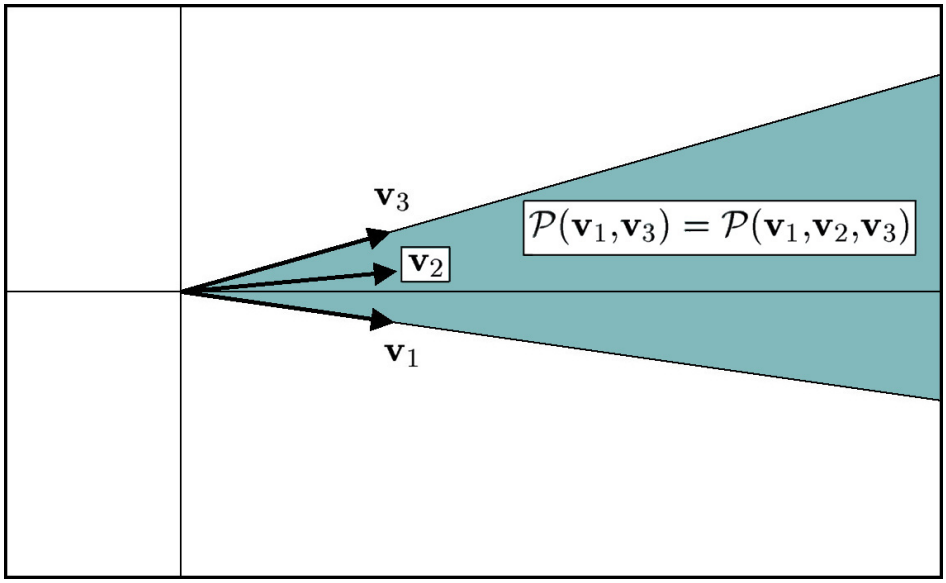


Fig. 4.6 – Cône positif engendré par trois vecteurs du plan. Le sous-ensemble minimal correspondant (unique dans ce cas), est composé des vecteurs $\mathbf{v}_1$ et $\mathbf{v}_3$.

Lemme 2. *Si* $\mathbf{n}_m \in \mathcal{P}(\tilde{\mathbf{n}}_1, \ldots, \tilde{\mathbf{n}}_r)$ *pour tout* m, $1 \leqslant m \leqslant k$, *alors* $\mathbf{CR}_v \subset \bigcap_{m=1}^{k} \mathbf{H}(\mathbf{n}_m)$.

Démonstration. Soit $\mathbf{x}$ un point dans $\mathbf{CR}_v$. Alors $\mathbf{x}.\tilde{\mathbf{n}}_m < 0$ pour tout m, $1 \leqslant m \leqslant r$. Pour tout m, $1 \leqslant m \leqslant k$, puisque $\mathbf{n}_m$ est une combinaison linéaire positive de $\tilde{\mathbf{n}}_1$, …, $\tilde{\mathbf{n}}_r$, et puisque le produit scalaire est bilinéaire, $\mathbf{x}.\mathbf{n}_m < 0$. □

L'hypothèse du lemme 2 peut être réinterprétée géométriquement. En effet :

$$(\mathbf{n}_m \in \mathcal{P}(\tilde{\mathbf{n}}_1, \ldots, \tilde{\mathbf{n}}_r), 1 \leqslant m \leqslant k) \Leftrightarrow (\mathcal{P}(\mathbf{n}_1, \ldots, \mathbf{n}_r) \subset \mathcal{P}(\tilde{\mathbf{n}}_1, \ldots, \tilde{\mathbf{n}}_r))$$

Par conséquent, le lemme 2 indique qu'un moyen de calculer un cône de recul est de trouver un cône positif *englobant* $\mathcal{P}(\mathbf{n}_1, \ldots, \mathbf{n}_r)$. Afin qu'un test de recul permette d'éliminer un volume englobant aussi souvent que possible, les r vecteurs de recul $\tilde{\mathbf{n}}_1$, …, $\tilde{\mathbf{n}}_r$ sont choisis de façon à minimiser la différence entre le cône de recul $\mathbf{CR}_v$ et $\bigcap_{m=1}^{k} \mathbf{H}(\mathbf{n}_m)$. Ainsi, le cône positif $\mathcal{P}(\tilde{\mathbf{n}}_1, \ldots, \tilde{\mathbf{n}}_r)$ doit approcher au mieux (en l'englobant), le cône positif $\mathcal{P}(\mathbf{n}_1, \ldots, \mathbf{n}_k)$. La figure 4.7 montre un raisonnement semblable dans $\mathbb{R}^2$.

Remarquons maintenant que, de par la propriété d'inclusion (4.8), il n'est intéressant de chercher à construire le cône que lorsque $\bigcap_{m=1}^{k} \mathbf{H}(\mathbf{n}_m)$ est non vide[2]. Le

[2]Lorsque $\bigcap_{m=1}^{k} \mathbf{H}(\mathbf{n}_m)$ est vide, le test de recul hiérarchique conclut toujours a un mouvement de recul. Dans une stratégie *top-down* d'ajout des cônes de recul, qui construit les vecteurs de recul à partir des normales aux triangles, le fait que $\bigcap_{m=1}^{k} \mathbf{H}(\mathbf{n}_m)$ soit vide signifie qu'il existe effectivement toujours au moins un triangle associé qui recule, quel que soit le mouvement de l'objet. Dans une stratégie *bottom-up*, cependant, $\bigcap_{m=1}^{k} \mathbf{H}(\mathbf{n}_m)$ peut être vide alors que certains triangles associés reculent parfois, à cause de la perte d'informations caractérisée par la relation d'inclusion (4.12).

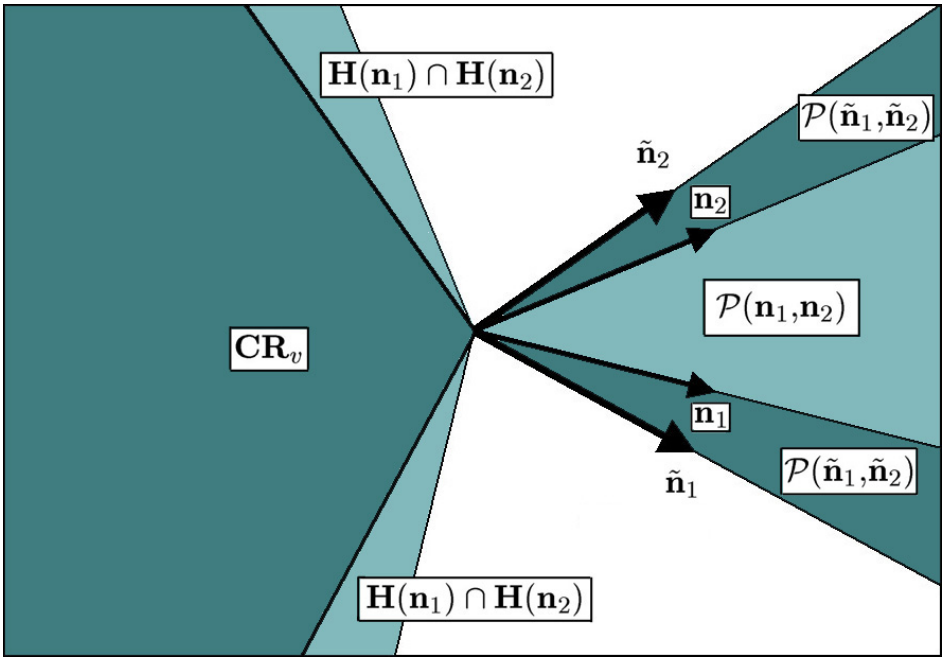


Fig. 4.7 – Détermination de vecteurs de recul satisfaisant la relation d'inclusion (4.8). En choisissant $\tilde{\mathbf{n}}_1$ et $\tilde{\mathbf{n}}_2$ de sorte que le cône positif $\mathcal{P}(\tilde{\mathbf{n}}_1, \tilde{\mathbf{n}}_2)$ englobe le cône positif $\mathcal{P}(\mathbf{n}_1, \mathbf{n}_2)$, le cône de recul $\mathbf{CR}_v$ est bien inclus dans $\mathbf{H}(\mathbf{n}_1) \cap \mathbf{H}(\mathbf{n}_2)$.

lemme suivant permet de savoir si le calcul d'un cône de recul peut être évité :

Lemme 3. *Supposons que* $\mathcal{P}(\mathbf{n}_1, \ldots, \mathbf{n}_k) = \mathbb{R}^3$, *alors* $\bigcap_{m=1}^{k} \mathbf{H}(\mathbf{n}_m)$ *est vide.*

Démonstration. Supposons qu'il existe un vecteur $\mathbf{x}$ dans $\mathbb{R}^3$ tel que $\mathbf{x}.\mathbf{n}_m < 0$, pour tout m, $1 \leqslant m \leqslant k$. Nécessairement, $\mathbf{x} \neq \mathbf{0}$. Puisque $\mathbf{x} \in \mathcal{P}(\mathbf{n}_1, \ldots, \mathbf{n}_k)$, il existe k valeurs positives $\lambda_1, \ldots, \lambda_k$ telles que $\mathbf{x} = \sum_{m=1}^{k} \lambda_m \mathbf{n}_m$. Comme $\mathbf{x}$ est non nul, l'une au moins de ces valeurs est strictement positive. Cependant, $||\mathbf{x}||^2 = \mathbf{x}.\mathbf{x} = \sum_{m=1}^{k} \lambda_m \mathbf{x}.\mathbf{n}_m < 0$. C'est impossible. □

Notons que ce lemme permet seulement d'éviter la construction de *la plupart* des cônes inutiles (vides). Parfois, les vecteurs $\mathbf{n}_1, \ldots, \mathbf{n}_k$ n'engendrent pas $\mathbb{R}^3$ alors que $\bigcap_{m=1}^{k} \mathbf{H}(\mathbf{n}_m)$ est vide, par exemple lorsque $\mathcal{P}(\mathbf{n}_1, \ldots, \mathbf{n}_k)$ contient une droite ou un plan mais *ne contient pas* $\mathbb{R}^3$. En de telles (rares) occasions, l'algorithme de construction détermine quand même un ensemble de r vecteurs $\tilde{\mathbf{n}}_1, \ldots, \tilde{\mathbf{n}}_r$. Cependant, le cône de recul résultant est vide.

Afin de déterminer en pratique quand $\mathcal{P}(\mathbf{n}_1, \ldots, \mathbf{n}_k)$ est égal à $\mathbb{R}^3$, l'algorithme de construction utilise un dernier lemme :

Lemme 4. $\mathcal{P}(\mathbf{n}_1, \ldots, \mathbf{n}_k) = \mathbb{R}^3$ *si et seulement si* $(\pm 1, 0, 0)$, $(0, \pm 1, 0)$ *et* $(0, 0, \pm 1)$ *sont dans* $\mathcal{P}(\mathbf{n}_1, \ldots, \mathbf{n}_k)$.

Démonstration. Notons $\mathbf{i} = (1,0,0)$, $\mathbf{j} = (0,1,0)$ et $\mathbf{k} = (0,0,1)$. Supposons que $\pm\mathbf{i}$, $\pm\mathbf{j}$ et $\pm\mathbf{k}$ sont dans $\mathcal{P}(\mathbf{n}_1, \ldots, \mathbf{n}_k)$, et notons $\mathbf{x} = x_i\mathbf{i} + x_j\mathbf{j} + x_k\mathbf{k}$. Supposons par exemple que $x_i \geqslant 0$, $x_j < 0$ et $x_k \geqslant 0$. Alors $\mathbf{x} = x_i\mathbf{i} + |x_j|(-\mathbf{j}) + x_k\mathbf{k}$. Puisque $\mathbf{i}$, $-\mathbf{j}$ et $\mathbf{k}$ sont dans $\mathcal{P}(\mathbf{n}_1, \ldots, \mathbf{n}_k)$, $\mathbf{x}$ est aussi une combinaison linéaire positive de $\mathbf{n}_1$, …, $\mathbf{n}_k$. Le raisonnement est le même pour les sept autres combinaisons de signes. □

4.5.2 Un algorithme de construction de cônes de recul

Interpréter les ensembles de vecteurs en termes de cônes positifs permet de mesurer la différence entre $\mathbf{CR}_v$ et $\bigcap_{m=1}^{k} \mathbf{H}(\mathbf{n}_m)$: dans l'algorithme de construction, cette différence est mesurée en calculant la distance des vecteurs de recul au cône $\mathcal{P}(\mathbf{n}_1, \ldots, \mathbf{n}_k)$.

Plus précisément, nous utilisons l'algorithme de Wilhelmsen [Wil76] pour calculer les projections $\tilde{\mathbf{n}}_1^* = \mathcal{W}(\tilde{\mathbf{n}}_1)$, …, $\tilde{\mathbf{n}}_r^* = \mathcal{W}(\tilde{\mathbf{n}}_r)$ des vecteurs de recul sur le cône $\mathcal{P}(\mathbf{n}_1, \ldots, \mathbf{n}_k)$. Les distances requises sont alors simplement $d_1 = ||\tilde{\mathbf{n}}_1 - \tilde{\mathbf{n}}_1^*||$, …, $d_r = ||\tilde{\mathbf{n}}_r - \tilde{\mathbf{n}}_r^*||$. Notons que l'algorithme de Wilhelmsen permet de savoir qu'un point appartient à un cône positif lorsque la distance entre ce point et son projeté est nulle.

Nous pouvons maintenant décrire un algorithme de construction de cônes de recul :

1. Test préliminaire
Calculer $\mathcal{W}(\pm\mathbf{i})$, $\mathcal{W}(\pm\mathbf{j})$ et $\mathcal{W}(\pm\mathbf{k})$. Si chaque point se projette sur lui-même, fin de l'algorithme. Aucun cône de recul ne peut être construit. Sinon, aller à l'étape 2.

2. Calcul du sous-ensemble minimal
Utiliser l'algorithme de Wilhelmsen en boucle pour supprimer des vecteurs linéairement dépendants des autres et obtenir un sous-ensemble minimal $\mathbf{v}_{i_1}$, …, $\mathbf{v}_{i_t}$[3]. Cette étape correspond à la figure 4.8.b. Si t est inférieur à r_{max}, alors les vecteurs de recul sont les vecteurs du sous-ensemble minimal. Sinon, aller a l'étape 3.

3. Initialisation du cône de recul
Calculer la moyenne des vecteurs du sous-ensemble minimaux $\mathbf{m} = \frac{1}{t}\sum_{j=1}^{t} \mathbf{v}_{i_j}$. Sélectionner aléatoirement r_{max} vecteurs $\tilde{\mathbf{n}}_1$, …, $\tilde{\mathbf{n}}_{r_{max}}$ parmi ceux du sous-ensemble minimal jusqu'à ce que $\mathbf{m}$ soit dans $\mathcal{P}(\tilde{\mathbf{n}}_1, \ldots, \tilde{\mathbf{n}}_{r_{max}})$ (r_{max} doit être supérieur à 3 afin que cela ne soit pas quasi impossible). Pousser les r_{max} vecteurs ainsi sélectionnés à l'extérieur du cône $\mathcal{P}(\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_t})$ (*i.e.* dans les directions $\tilde{\mathbf{n}}_i - \mathbf{m}$). Cette étape est représentée dans la figure 4.8.c.

4. Raffinement du cône de recul
Raffiner progressivement le cône de recul par une méthode de recuit simulé

[3]Nous proposons d'utiliser l'algorithme de Wilhelmsen à cette étape afin de minimiser le temps de développement global de l'algorithme. Pour une implantation à plus grande échelle, nous suggérons d'adapter un algorithme de calcul d'enveloppe convexe.

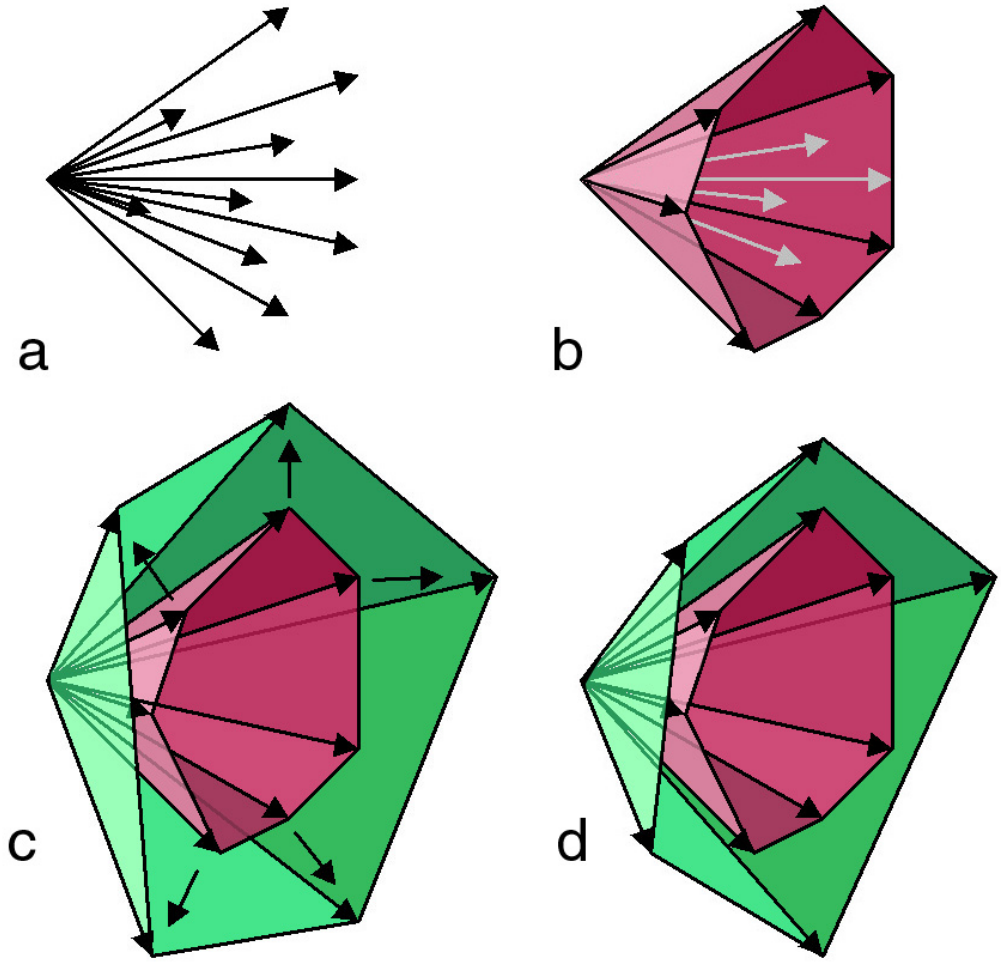


Fig. 4.8 – Etapes principales pour construire des vecteurs de recul à partir d'un ensemble de vecteurs. a : l'ensemble de vecteurs $\mathbf{n}_1, \ldots, \mathbf{n}_k$. b : détermination d'un sous-ensemble minimal $\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_t}$. c : initialisation des vecteurs de recul $\tilde{\mathbf{n}}_1, \ldots, \tilde{\mathbf{n}}_{r_{max}}$. d : raffinement du cône de recul.

[PTVF92]. Brièvement, les vecteurs de recul sont légèrement modifiés étape par étape. Les modifications sont valides si la contrainte d'inclusion (4.8) est respectée et si le taux d'accroissement des distances $d_1, \ldots, d_{r_{max}}$ est inférieur à une valeur prédéterminée. L'algorithme de Wilhelmsen est utilisé pour vérifier la contrainte d'inclusion (grâce au lemme 2) et pour calculer les distances. Cette étape correspond à la figure 4.8.d.

Nous verrons à la section 4.7 que cet algorithme permet d'ajouter les cônes de recul aux volumes englobants en un temps raisonnablement court, en particulier lorsque la stratégie *bottom-up* est utilisée.

4.6 Tests de recul hiérarchiques continus

Pour l'instant, la méthode présentée ne dépend que des vitesses relatives à un instant donné. Bien que ceci soit valide pour des techniques de détection de collisions discrètes, il est nécessaire d'étendre la méthode aux techniques continues. Un triangle peut en effet reculer au début du mouvement et ne pas reculer sur l'ensemble de l'intervalle de temps considéré.

Cette extension est simple en utilisant l'arithmétique d'intervalles (*cf* chapitre 3). Il suffit en effet de borner les produits scalaires dans les tests d'éliminations (4.9) :

$$\dot{\mathbf{c}}_{ij}^{s}.\tilde{\mathbf{n}}_m \in [l_{s,m}, u_{s,m}] \tag{4.13}$$

Le test de recul hiérarchique est alors semblable à celui de la section 4.3 : un volume englobant est éliminé si $u_{s,m} < 0$, pour tout s, $1 \leqslant s \leqslant t$, et pour tout m, $1 \leqslant m \leqslant r$.

Notons qu'ici aussi les tests continus bénéficient de l'utilisation d'un mouvement intermédiaire arbitrairement fixé, qui permet d'implanter *en dur* les calculs d'intervalles et, ainsi, de les optimiser.

Bien sûr, utiliser l'arithmétique d'intervalles pour borner les produits scalaires diminue légèrement l'efficacité des tests, qui deviennent un peu plus conservatifs encore. En effet, les bornes obtenues se sont pas nécessairement les meilleures. Ceci pourrait être amélioré en subdivisant l'intervalle de temps considéré, comme nous l'avons fait au chapitre précédent pour le test de recouvrement continu entre boîtes englobantes. Nous avons cependant jugé que cette amélioration n'aurait été qu'assez faible, au regard de l'accélération déjà très significative permise par l'utilisation des cônes de recul, comme le montre la section suivante.

4.7 Evaluation des tests de recul hiérarchiques

4.7.1 Accélération de la détection de collisions

Afin de tester la méthode des cônes de recul, nous l'avons intégrée aux algorithmes de détection de collisions continue présentés au chapitre précédent, qui combinent l'arithmétique d'intervalles et les boîtes englobantes orientées, et aux algorithmes de simulation dynamique présentés dans le chapitre suivant.

Le test a consisté à effectuer un mouvement qui constitue un cas difficile pour toute méthode de détection de collisions utilisant des hiérarchies de volumes englobants (c'est-à-dire la plupart des méthodes permettant de détecter des collisions entre des objets aux topologies quelconques). Dans ce test, deux portières identiques à celle de la figure 4.5 ont été utilisées. L'une d'elle, mobile, était manipulée par un utilisateur qui l'amenait en contact avec la deuxième, statique. Une fois en contact, plusieurs

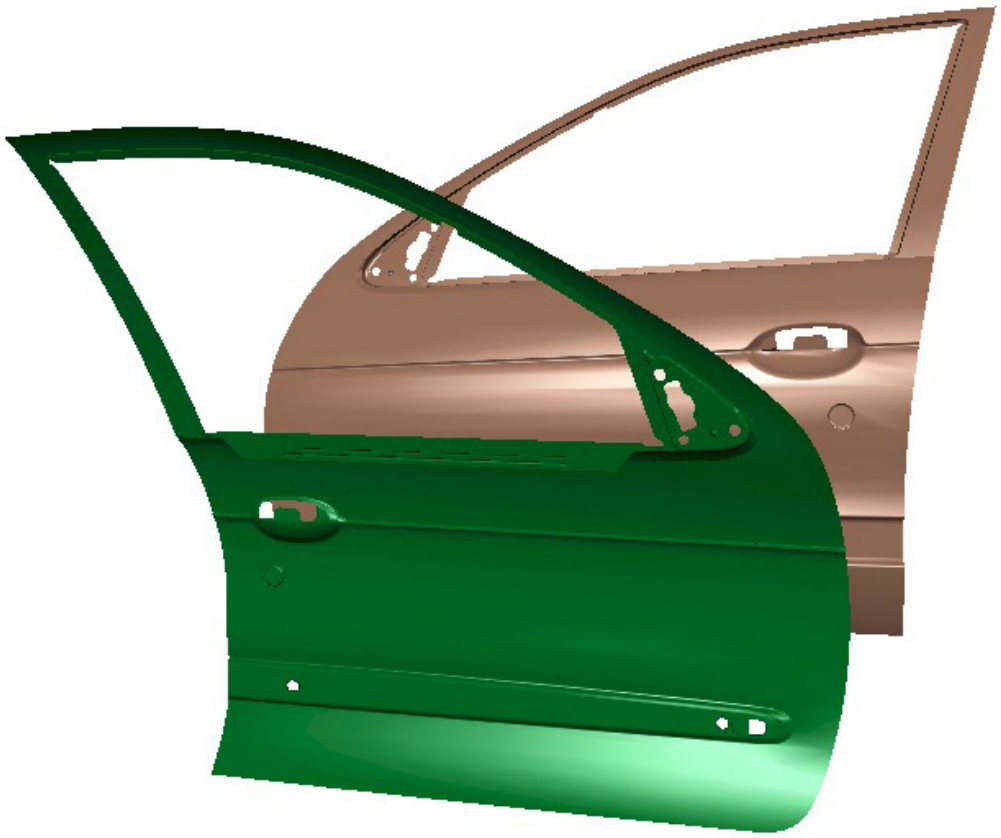


Fig. 4.9 – Extrait de la trajectoire de test : la porte mobile, manipulée par l'utilisateur, glisse sur la porte statique et entraîne de nombreux tests de détection de collisions.

mouvements de glissement, incorporant des translations et des rotations étaient réalisés. La figure 4.9 montre une image extraite de la simulation.

Les mouvements de l'utilisateur, enregistrés, étaient ensuite rejoués dans trois configurations différentes :

1. sans les tests de recul,
2. avec les tests de recul élémentaires lorsque des volumes englobants feuilles se recouvrent (*i.e.* l'extension directe de la méthode de Vanecek),
3. avec les tests de recul hiérarchiques.

Pour les tests de recul hiérarchiques, le nombre maximal de vecteurs de recul par cône (r_{max}) était fixé à 6, et les cônes ajoutés grâce à la méthode *top-down*.

Le tableau 4.1 présente les statistiques de la simulation dans chaque cas. N_{TR} et T_{TR} sont respectivement le nombre de tests de recul (élémentaires ou hiérarchiques) et le temps total passé dans les fonctions de test de recul. L'entier f est le taux de rafraîchissement moyen de la simulation (en images par seconde), et α est l'accélération

	Sans tests de recul	Tests de recul élémentaires	Tests de recul hiérarchiques
N_{BV}	7.483.916	6.197.548	4.860.450
N_{AA}	6.290.371	2.968.201	2.945.669
N_{PF}	2.068.579	971.035	966.109
N_{FP}	2.062.241	968.633	965.067
N_{TR}	-	219.935	521.290
T_{BV}	32,42s	26,58s	20,82s
T_{AA}	4,99s	3,22s	2,70s
T_{PF}	2,67s	1,45s	1,48s
T_{FP}	2,68s	1,50s	1,43s
T_{TR}	-	0.78s	1,52s
T	49,81s	39,31s	32,69s
f	48	61	73
α	**-**	**+27%**	**+52%**

Tab. 4.1 – Evaluation des tests de recul hiérarchiques. Les tests de recul hiérarchiques permettent de réduire le nombre de tests de recouvrement entre volumes englobants et le nombre de détection de collisions élémentaires, ce qui conduit à une amélioration du taux de rafraichissement de plus de 50%.

moyenne de la simulation, représentée par le taux d'accroissement du taux de rafraîchissement. Ces statistiques montrent que si l'extension naturelle de Vanecek permet déjà une accélération moyenne de 27% environ, les tests de recul hiérarchiques réalisent une accélération de la détection de collisions de plus de 50%, grâce à une réduction importante du nombre de tests de recouvrement entre volumes englobants et de tests de détection de collisions entre primitives polyédriques.

Il faut noter l'importance des entiers s_A^t et s_B^t , utilisés dans l'algorithme adapté de parcours des hiérarchies englobantes (*cf* section 4.4.1), et permettant de savoir si les volumes englobants A ou B ont déjà subi un test de recul hiérarchique. Il apparaît en effet que, sans cet entier, le nombre de tests de recul effectué est près de quatre fois plus important (dans l'exemple choisi, N_{TR} passe de 521.290 à 2.136.234, et T_{TR} passe de 1,52s à 5,85s).

4.7.2 Influence de r_{max} et de la stratégie d'ajout des cônes

Afin de détailler les résultats précédents, nous avons testé l'influence de la constante r_{max} et de la stratégie utilisée pour ajouter les cônes (*top-down* ou *bottom-up*) sur l'accélération de la détection de collisions. Pour cela, nous avons utilisé le même mouvement que précédemment et mesuré à chaque fois le temps total passé dans les fonctions de simulations. Le tableau 4.2 donne les résultats en fonction de la constante r_{max} pour chacune des deux stratégies. Selon la stratégie de construction employée,

r_{max}	3	4	5	6	7	8	9
$T_{bottom-up}$ (secondes)	35,03	35,54	34,20	34,22	34,06	33,92	33,69
$\alpha_{bottom-up}$(%)	+42,2	+40,2	+45,6	+45,6	+46,2	+46,8	+47,8
$T_{top-down}$ (secondes)	33,83	33,31	33,12	32,69	32,63	32,17	31,88
$\alpha_{top-down}$ (%)	+47,2	+49,5	+50,4	+52,4	+52,7	+54,8	+56,2

Tab. 4.2 – Influence de r_{max} et de la stratégie d'ajout des cônes sur l'accélération. Quelle que soit la stratégie choisie pour ajouter les cônes dans les volumes englobants, l'accélération de la détection de collisions est d'autant plus importante que la constante r_{max} est grande, car les cônes de recul approchent de mieux en mieux les normales aux triangles contenus dans les volumes englobants, et permettent d'éliminer des volumes englobants qui reculent de plus en plus précisément (le test devient de moins en moins conservatif et de plus en plus exact).

ce temps total, équivalent au temps T de la section précédente, est noté $T_{bottom-up}$ ou $T_{top-down}$. Les accélérations correspondantes $\alpha_{bottom-up}$ et $\alpha_{top-down}$ sont aussi données.

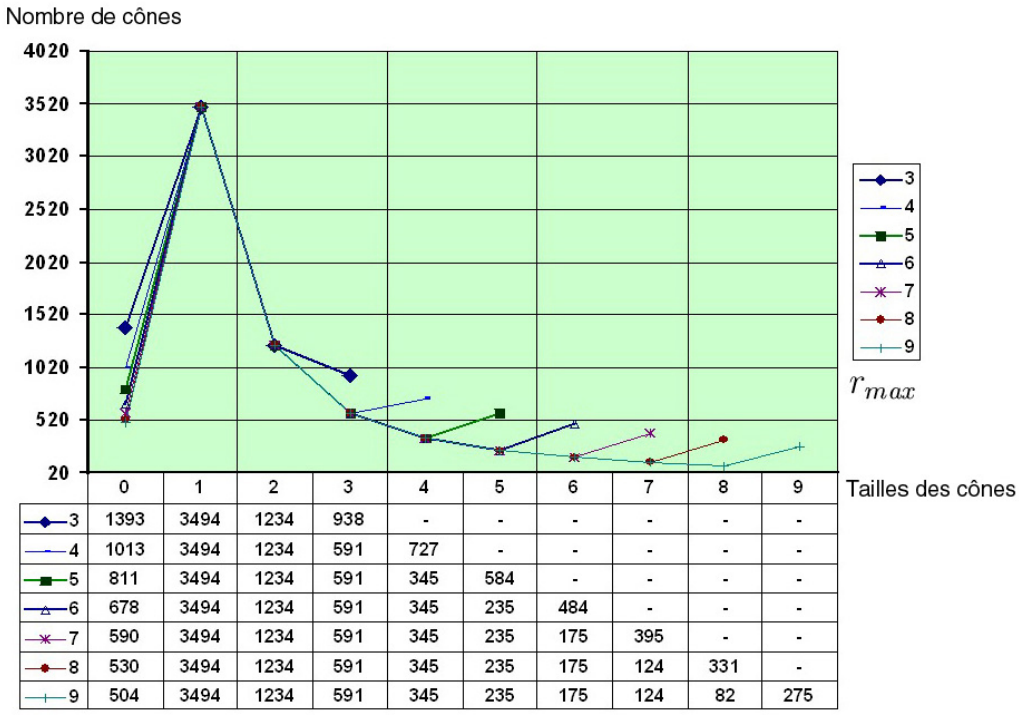
Il apparaît clairement qu'augmenter la précision des cônes de recul (en augmentant r_{max}) permet d'améliorer l'efficacité des tests hiérarchiques, et ce malgré l'accroissement de leur coût propre. Aussi, pour une taille maximale r_{max} donnée, la stratégie *top-down* conduit à une accélération plus importante que la stratégie *bottom-up*. Pour les deux stratégies, cependant, l'accélération résultante est bien supérieure à celle permise par l'extension directe de la méthode de Vanecek.

4.7.3 Encombrement mémoire

Il est difficile, voire impossible, de prédire l'encombrement mémoire pour un objet spécifique, puisque les cônes de recul dépendent fortement de la géométrie de l'objet, de la façon de construire les hiérarchies englobantes, et de la façon d'ajouter les cônes aux volumes englobants.

Il est cependant facile de *mesurer* a posteriori le coût des cônes de recul, en parcourant la hiérarchie englobante d'un objet. Les figures 4.10 et 4.11 montrent la répartition des tailles des cônes de recul dans les hiérarchies englobantes, en fonction de la taille maximale autorisée r_{max}, pour la portière utilisée dans le test. Le tableau 4.10 donne les répartitions lorsque la stratégie *top-down* est utilisée, alors que le tableau 4.11 donne les répartitions pour la stratégie *bottom-up*.

De manière générale, il semble que la taille moyenne des cônes produits par la stratégie *top-down* soit plus élevée. Ceci est essentiellement dû à la deuxième étape de l'algorithme de construction d'un cône, qui calcule un sous-ensemble minimal des vecteurs à partir desquels le cône doit être construit. Si le nombre de vecteurs du sous-ensemble minimal est inférieur à la constante r_{max}, il n'est pas nécessaire d'aller à la troisième étape : les vecteurs de recul sont les vecteurs du sous-ensemble minimal. Ceci



	0	1	2	3	4	5	6	7	8	9
3	1393	3494	1234	938	-	-	-	-	-	-
4	1013	3494	1234	591	727	-	-	-	-	-
5	811	3494	1234	591	345	584	-	-	-	-
6	678	3494	1234	591	345	235	484	-	-	-
7	590	3494	1234	591	345	235	175	395	-	-
8	530	3494	1234	591	345	235	175	124	331	-
9	504	3494	1234	591	345	235	175	124	82	275

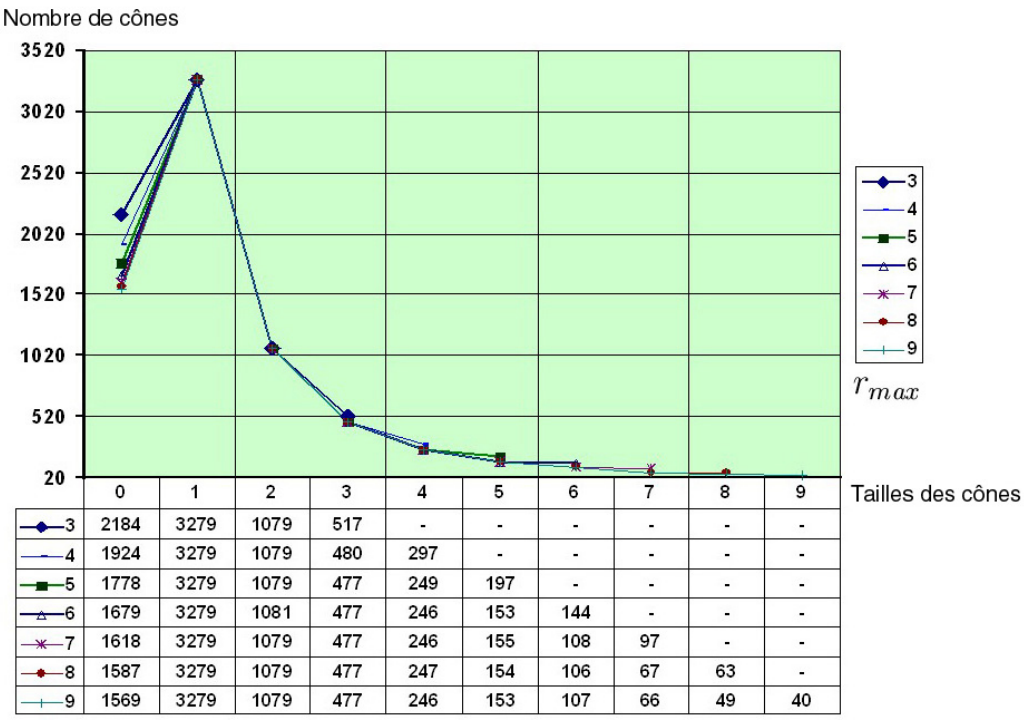
Fig. 4.10 – Méthode *top-down*. Dénombrement des cônes de recul dans la hiérarchie englobante en fonction de leur taille (en nombre de vecteurs de recul) et de la taille maximale r_{max} autorisée pour un cône.

se produit beaucoup plus souvent dans une stratégie de construction *bottom-up*, qui construit les cônes des noeuds internes à partir des cônes des descendants directs : dans une hiérarchie englobante binaire (*i.e.* pour laquelle chaque noeud a au plus deux fils), le nombre de vecteurs servant à construire le cône de recul est en permanence inférieur ou égal à $2 \times r_{max}$, et les chances qu'un sous-ensemble minimal de ces vecteurs contienne peu de vecteurs sont plus grandes.

C'est d'ailleurs cette deuxième étape de l'algorithme de construction qui explique que les deux stratégies produisent plus de cônes de taille r_{max} que de cônes de taille $r_{max} - 1$: parmi les cônes de taille r_{max}, certains ont été obtenus en éliminant des vecteurs d'un sous-ensemble minimal qui contenait trop de vecteurs. Ce phénomène est très visible dans la stratégie *top-down*, qui construit naturellement des sous-ensembles minimaux de grande taille.

Dans notre implantation, les cônes de recul sont ajoutés aux volumes englobants sous la forme de pointeurs vers des structures (afin d'économiser de la place lorsque le cône de recul est vide), et la structure d'un cône de recul est :

```
struct Cone {
```



	0	1	2	3	4	5	6	7	8	9
3	2184	3279	1079	517	-	-	-	-	-	-
4	1924	3279	1079	480	297	-	-	-	-	-
5	1778	3279	1079	477	249	197	-	-	-	-
6	1679	3279	1081	477	246	153	144	-	-	-
7	1618	3279	1079	477	246	155	108	97	-	-
8	1587	3279	1079	477	247	154	106	67	63	-
9	1569	3279	1079	477	246	153	107	66	49	40

Fig. 4.11 – Méthode *bottom-up*. Dénombrement des cônes de recul dans la hiérarchie englobante en fonction de leur taille (en nombre de vecteurs de recul) et de la taille maximale r_{max} autorisée pour un cône.

```
    int taille; // taille du cone
    double *cone; // tableau des 3*taille coordonnees des vecteurs de recul
    int st; // pour savoir si le BV a deja subi un test de recul
    int se; // pour savoir si le volume englobant a deja ete elimine

}
```

Notons que, dans notre implantation fondée sur les boîtes englobantes orientées, il n'est pas nécessaire de stocker les points caractéristiques, qui peuvent être calculés pendant le test de recul. Pour d'autres volumes englobants, il peut être nécessaire de précalculer les points caractéristiques qui sont alors stockés dans la structure du cône de recul (puisqu'il est inutile de les mémoriser lorsque le cône est vide).

De manière générale, l'encombrement mémoire total dû à l'utilisation des tests hiérarchiques est donc :

$$\mathcal{M} = C_{VR} \times \sum_{i=1}^{r_{max}} (i \times N_i) + (C_s + C_c) \times \sum_{i=1}^{r_{max}} N_i + N_v \times C_p \qquad (4.14)$$

où N_i est le nombre de cônes de taille i, C_{VR} l'encombrement d'un vecteur de recul, C_s l'encombrement des trois entiers contenus dans la structure du cône, C_c l'encombrement

dû à l'ajout des points caractéristiques dans le cône de recul, N_v le nombre de volumes englobants dans la hiérarchie englobante, et C_p l'encombrement d'un pointeur vers une structure de cône.

4.7.4 Temps de construction des cônes de recul

En raison de la complexité de l'algorithme de Wilhelmsen, polynomiale dans le cas le pire, la stratégie *top-down* est beaucoup plus longue que la stratégie *bottom-up*. Le tableau 4.3 donne le temps total d'ajout des cônes en fonction de la constante r_{max} et de la stratégie employée.

r_{max}	3	4	5	6	7	8	9
$T^c_{bottom-up}$ (secondes)	6,31	13,86	12,05	13,10	14,71	10,43	9,73
$T^c_{top-down}$ (secondes)	137,2	175,3	192,6	212,8	219,8	233,2	231,3

Tab. 4.3 – Evolution du temps total pour ajouter les cônes en fonction de r_{max}.

Il semble ainsi préférable d'utiliser une stratégie *bottom-up* lorsque la complexité des objets augmente. Nous avons vu que si l'accélération de la détection de collisions est moins importante pour une stratégie *bottom-up* que pour une stratégie *top-down* pour une valeur de r_{max} donnée, elle reste néanmoins très significative.

De plus, les objets étant rigides, les cônes de calcul précalculés peuvent être stockés dans les fichiers décrivant les objets et les hiérarchies englobantes.

4.8 Conclusion

Dans ce chapitre, nous avons présenté une technique d'accélération de la détection de collisions, la méthode des *tests de recul hiérarchiques*, fondée sur l'exploitation du *mouvement* des objets et non plus seulement de leur position, comme le font les volumes englobants. Cette technique, généralisation d'une méthode de Vanecek [Van94], consiste à ajouter des *cônes de recul* dans les volumes englobants afin d'approcher les normales des triangles qui leur sont associés. De par leur taille bornée, ces cônes permettent de détecter en temps constant la plupart des situations où tous les triangles associés à un volume englobant reculent, autrement dit les situations dans lesquelles il n'est pas nécessaire de parcourir la descendance du volume englobant, et ce même lorsqu'il recouvre un volume englobant d'un autre objet.

Nous avons proposé un algorithme de construction de cônes de recul, ainsi que deux stratégies pour ajouter les cônes de recul aux hiérarchies englobantes. Des tests ont permis de montrer que de ces deux stratégies, la stratégie *top-down*, qui construit le cônes de recul d'un volume englobant directement à partir des normales aux triangles

qui lui sont associés, est à la fois la plus longue à calculer, la plus encombrante en mémoire, mais aussi celle qui permet d'accélérer le plus la détection de collisions.

Ainsi, des tests ont montré que les tests de recul hiérarchiques accélèrent la détection de collisions très significativement lorsque les objets sont proches les uns des autres, puisqu'elle permet d'éviter de nombreux de tests de recouvrement entre volumes englobants, ainsi que de nombreux tests de détection de collisions entre primitives polyédriques. Dans l'exemple des portières en contact, représentant un cas difficile pour toute méthode de détection de collisions fondée sur des hiérarchies englobantes, choisir une stratégie *top-down* et fixer la taille maximale r_{max} à 6 permet d'augmenter le taux de rafraîchissement de la simulation de plus de 50%.

La stratégie *bottom-up*, cependant, n'est pas à négliger. Bien qu'elle permette une accélération un peu moins importante que la stratégie *top-down*, elle requiert légèrement moins de mémoire et, surtout, calcule les cônes de recul beaucoup plus rapidement. Aussi la préférons-nous lorsque la complexité des objets augmente.

La méthode proposée autorise également un compromis entre la mémoire requise par les cônes de recul et l'accélération résultant de leur utilisation. Ici aussi des tests ont permis d'évaluer cet encombrement mémoire en fonction de la stratégie d'ajout des cônes de recul d'une part, et de la constante r_{max}, qui définit la taille maximale d'un cône de recul au sein d'une hiérarchie, d'autre part.

Dans le chapitre suivant, nous allons nous intéresser au deuxième problème fondamental de la simulation interactive d'objets rigides, le calcul du mouvement des objets. Nous avons vu au chapitre 1 que ce problème est lui-même divisé en deux sous-problèmes selon que l'on doive calculer les *vitesses* des objets, après qu'un choc les ait rendues incompatibles avec les contraintes géométriques, ou les *accélérations des objets* qui, en l'absence de chocs, doivent respecter continûment les contraintes géométriques.

Essentiellement, nous allons montrer que si les méthodes analytiques traditionnelles, qui résolvent les problèmes dynamiques dans *l'espace des contacts*, permettent à l'heure actuelle de simuler interactivement des objets rigides, elles ne sont pas aussi efficaces qu'elles pourraient l'être. Nous allons en effet montrer que le principe de Gauss, qui permet de résoudre les problèmes dynamiques dans *l'espace des mouvements*, permet d'obtenir une formulation mathématiquement équivalente à celle de l'approche traditionnelle, mais algorithmiquement avantageuse.

Chapitre 5

Simulation dynamique dans l'espace des mouvements

La plupart des méthodes classiques de simulation dynamique d'objets rigides par contraintes sont formulées dans l'espace des contacts. Grâce au principe des moindres contraintes de Gauss, les problèmes dynamiques sans friction sont formulés dans l'espace des mouvements. Bien que les deux formulations soient mathématiquement équivalentes, elles ne le sont pas d'un point de vue algorithmique. La formulation établie dans l'espace des mouvements est mieux conditionnée, toujours creuse, nécessite moins de mémoire et permet d'éviter certains calculs redondants. Une comparaison expérimentale montre qu'un algorithme opérant dans l'espace des mouvements tire parti de la formulation creuse pour résoudre les problèmes dynamiques de plus en plus efficacement (par rapport à la formulation dans l'espace des contacts) à mesure que le nombre moyen de contacts par objet augmente.

5.1 Introduction

Les chapitres précédents se sont concentrés sur le développement de méthodes de détections de collisions continues suffisament efficaces pour interagir avec un environnement virtuel en temps réel. Ainsi, contrairement à la plupart des méthodes de détection de collisions, qui détectent des *interpénetrations* entre les objets, les méthodes présentées dans les chapitres 2 et 3 permettent de calculer efficacement les instants auxquels les objets entrent en contact. Par ailleurs, le chapitre précédent nous a permis de montrer que le mouvement des objets lui-même pouvait être exploité afin d'accélérer la détection de collisions, grâce à la méthode des tests de recul hiérarchiques.

Nous avons vu au chapitre 1 qu'une fois l'instant de premier contact entre deux objets connu, il faut calculer la réponse à la collision (lorsque les objets doivent rebondir l'un sur l'autre), ou, de façon semblable, calculer le mouvement contraint des objets (lorsque les objets glissent l'un sur l'autre).

Si les premières méthodes de simulation dynamique, les méthodes par *pénalité*, calculaient simplement les forces exercées sur les objets en fonction des quantités d'interpénétration *locales* [MW88, TPB87, PB88], il s'est avéré que de telles méthodes conduisent généralement à des problèmes de stabilité numérique, et sont de plus difficilement justifiables d'un point de vue mécanique, notamment lorsque plusieurs points de contact coexistent [Bar89]. Ainsi, dans le cas d'une table posée sur le sol, la force totale exercée par le sol dépend du nombre de pieds de la table, ce qui est bien sûr physiquement incorrect.

Aussi la plupart des méthodes de simulation sont-elles maintenant *analytiques*, et calculent les forces exercées sur les objets en prenant en compte simultanément toutes les contraintes exercées sur les objets, qu'elles soient unilatérales ou bilatérales. A ce titre, les méthodes analytiques peuvent être qualifiées de *globales*. Dans l'exemple de la table, elles calculent le mouvement correct quel que soit le nombre de pieds de la table. Nous avons décrit le fonctionnement général d'un tel simulateur dynamique par contrainte dans le premier chapitre.

Bien que des objets modérément complexes puissent être gérés en temps réel par les processeurs et les méthodes de simulation actuels [Bar94], les objets complexes peuvent nécessiter de nombreux calculs lorsque le nombre de points de contact augmente, et une stratégie consiste souvent à *afficher* des objets complexes tout en effectuant la simulation dynamique sur des versions simplifiées des objets [FTBAB00].

Nous pensons qu'une des raisons de ce problème est que ces méthodes de simulation analytiques n'utilisent pas explicitement le nombre de degrés de liberté dans la simulation[1]. Aussi, lorsque le nombre moyen de contacts par objet augmente, les simulateurs ne sont pas aussi efficaces qu'ils pourraient l'être.

Notons que le nombre moyen de contacts par objet peut augmenter rapidement lorsque la complexité des objets augmente. Considérons deux cubes en contact. Si les cubes sont presque alignés et que l'un d'entre eux est légèrement tourné, alors huit points de contact coexistent (*cf* figure 5.1). Si maintenant les deux cubes sont remplacés par des cylindres polyédriques à n côtés dans une configuration semblable, alors $2n$ *contacts coexistent*.

D'autres exemples sont faciles a trouver : insertion de vis, objets empilés... Aussi, le fait que des valeurs de tolérance soient utilisées en détection de collisions tend à augmenter le nombre moyen de points de contacts par objet.

Enfin, notons qu'il n'est possible de réduire la zone de contact à un nombre fini de points caractéristiques (comme dans l'exemple de la figure 5.1) que lorsque les objets sont polyédriques [Pal87]. Dans le cas d'objets plus généraux, comportant des parties courbes, il arrive parfois qu'une infinité de points soit nécessaire pour décrire la zone de contact. C'est le cas par exemple lorsqu'un cylindre de section circulaire est polsé sur le sol. Dans ce cas, il est nécessaire de recourir à un grand nombre de points de contact pour approcher la surface de contact.

[1] Bien sûr, la détection de collisions est aussi un important facteur de ralentissement.

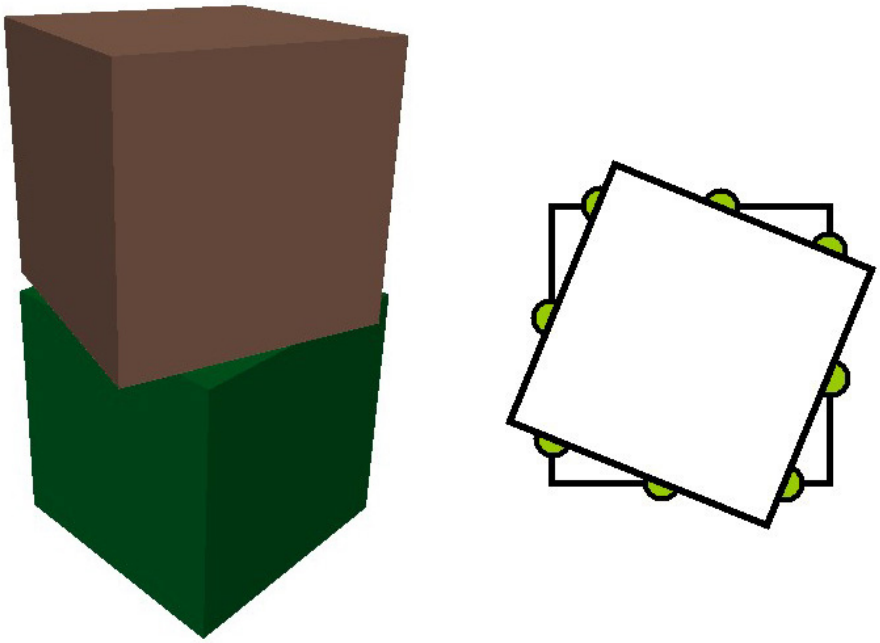


Fig. 5.1 – Selon le modèle de contact typiquement utilisé pour des objets polyédriques [Pal87], deux cubes en contact créent huit points de contact.

La plupart des méthodes de simulation dynamique analytiques permettant de gérer les contraintes unilatérales formulent les problèmes dynamiques (*i.e.* le *problème du calcul du mouvement contraint* et le *problème de la réponse à la collision*) comme des problèmes de complémentarité linéaire (*Linear Complementarity Problem*, LCP) (voir par exemple [APS99, Bar94, RK97, SS98a, TPSL95], et les méthodes citées dans le premier chapitre). Nous avons vu dans le premier chapitre que le LCP exprime la relation entre les forces de contact (respectivement les impulsions) et les accélérations normales relatives (respectivement les vitesses normales relatives) aux points de contact. Par exemple, dans un système sans frottements, si $\mathbf{f}$ et $\mathbf{a}_{cp}$ sont deux vecteurs dans $\mathbb{R}^m$ décrivant les composantes normales des forces de contact[2] et les composantes normales des accélérations relatives aux m points de contact, alors il existe une matrice $\mathbf{A}$ de taille $m \times m$ et un vecteur $\mathbf{b}$ de $\mathbb{R}^m$ tels que :

$$\mathbf{a}_{cp} = \mathbf{A}\mathbf{f} + \mathbf{b}, \;\; \mathbf{a}_{cp} \geqslant 0, \;\; \mathbf{f} \geqslant 0, \;\; \mathbf{a}_{cp}^T \mathbf{f} = 0 \tag{5.1}$$

Dans cette formulation, cependant, le nombre de degrés de liberté du système non contraint n'est pas explicite. En d'autres termes, le problème (5.1) est formulé dans l'espace des contacts [RK97].

Dans ce chapitre, nous présentons une formulation des deux problèmes dynamiques dans *l'espace des mouvements*, obtenue à partir du principe des moindres

[2]Dans un système sans frottements, les composantes tangentielles des forces de contact sont nulles.

contraintes de Gauss. Nous allons pouvoir observer que, dans cet espace, un algorithme résolvant les problèmes dynamiques est capable d'utiliser explicitement le nombre de degrés de liberté dans la simulation et d'éviter ainsi des calculs redondants.

Notons que certains auteurs ont présenté des formulations qui contiennent explicitement le nombre de degrés de liberté de la simulation. Lötstedt [Lot84] emploie une formulation très proche du principe de Gauss, mais ne prend pas en compte les collisions élastiques. Baraff [Bar96] présente un algorithme en temps linéaire pour les corps articulés acycliques, mais utilise ensuite une formulation dans l'espace des contacts [Bar94] pour gérer les contraintes unilatérales. Milenkovic [MS01], aussi, rend le nombre de degrés de liberté explicite en formulant les problèmes dynamiques comme des problèmes de programmation quadratique, mais a recours à de nombreuses variables pour faire respecter toutes les contraintes de dynamique et de collisions[3].

Ce chapitre est organisé de la façon suivante. Le principe de Gauss est rappelé dans la section 5.2. Une formulation dans l'espace des mouvements des deux problèmes dynamiques (sans frottements) est dérivée du principe de Gauss dans la section 5.3 pour former des problèmes de minimisation semblables. La section 5.4 réduit le problème de minimisation à un problème bien connu de recherche de point le plus proche (*Nearest Point Problem*, NPP), qui sera comparé à la formulation établie dans l'espace des contacts dans la section 5.5 d'un point de vue théorique et pratique. Cette comparaison montre que, bien que les deux formulations sont *mathématiquement* équivalentes, elles ne le sont pas d'un point de vue *algorithmique* : la formulation opérant dans l'espace des mouvements est mieux conditionnée, toujours creuse, nécessite moins de mémoire et permet d'éviter des calculs redondants. Une comparaison expérimentale est ensuite établie entre deux algorithmes typiquement utilisés pour résoudre les problèmes LCP et NPP. Cette comparaison montre que l'algorithme opérant dans l'espace des mouvements (*i.e.* résolvant le problème NPP) tire parti de la formulation creuse pour résoudre les problèmes dynamiques de plus en plus efficacement (par rapport à la formulation dans l'espace des contacts) à mesure que le nombre moyen de contacts par objet augmente. La section 5.6 propose un moyen d'étendre la formulation aux systèmes avec frottements, en présentant un modèle de friction inspiré du modèle de Coulomb, incluant la friction statique et dynamique. Enfin, la section 5.7 conclut le chapitre.

5.2 Le principe des moindres contraintes de Gauss

Il existe peu de références sur les applications du principe de moindres contraintes de Gauss [Gauss1829]. Ce principe a cependant récemment soulevé un regain d'intérêt dans les ouvrages généraux de mécanique [Bar99, UK96].

[3] De plus, un examen attentif du modèle employé pour calculer la réponse à la collision montre que celle-ci ne dépend pas de la vitesse de l'objet avant impact, ce qui est peu réaliste.

Plus spécifiquement, le principe de Gauss a été utilisé dans le cas particulier d'objets initialement immobiles [BM93] et, en robotique, pour calculer la dynamique de manipulateurs redondants [BK00]. Udwadia [UK96] montre que le principe de Gauss permet de formuler explicitement le mouvement des objets lorsque le système ne contient que des contraintes bilatérales. Un avantage immédiat du principe de Gauss sur le principe des travaux virtuels, utilisé dans les méthodes LCP, est qu'il permet de donner une formulation très intuitive du mouvement contraint d'un système. En fait, il est si intuitif qu'il est souvent redécouvert et/ou utilisé sans même mentionner Gauss ([MS01]).

Bien qu'il ait été initialement exprimé pour un ensemble de masses ponctuelles, le principe de Gauss s'applique aussi aux systèmes d'objets rigides sans frottements soumis à des contraintes géométriques (unilatérales ou bilatérales) [Mor63, Bar99, LL82], et peut être exprimé simplement à l'aide des coordonnées et des masses généralisées [Fea87] et de la notion de *groupe de contacts* :

Définition (Groupe de contacts). *Dans ce chapitre, un groupe de contacts est un ensemble d'objets mobiles en contact. Deux objets i et j sont dans le même groupe de contacts si et seulement s'il existe une chaîne d'objets en contact de l'objet i à l'objet j.*

Dans la figure 3.7 du chapitre 3, par exemple, les six cubes mobiles forment un seul et même groupe de contacts.

A chaque instant, deux groupes de contacts distincts sont dynamiquement indépendants. Les problèmes dynamiques qui leur sont associés peuvent donc être résolus indépendamment.

Considérons un groupe de contacts de n objets mobiles. Un objet rigide i a seulement six degrés de liberté, et son accélération est répartie en deux termes : l'accélération *translationnelle* $\mathbf{a}_i(G_i)$, qui est celle de son centre de gravité, et l'accélération *rotationnelle* α_i. Ces deux vecteurs sont dans $\mathbb{R}^3$. En empilant dans un même vecteur les accélérations des n objets (potentiellement) mobiles, on obtient le vecteur d'accélération généralisée $\mathbf{a}$ dans $\mathbb{R}^{6n}$:

$$\mathbf{a} = \begin{pmatrix} \mathbf{a}_i(G_1) \\ \alpha_1 \\ \vdots \\ \mathbf{a}_i(G_n) \\ \alpha_n \end{pmatrix} \tag{5.2}$$

De même, on forme la matrice des masses généralisées à partir des masses m_i des

objets et de leurs tenseurs d'inertie $\mathbf{I}_i$:

$$\mathbf{M} = \begin{pmatrix} m_1\mathbf{1} & 0 & \dots & 0 & 0 \\ 0 & \mathbf{I}_1 & \dots & 0 & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & \dots & m_n\mathbf{1} & 0 \\ 0 & 0 & \dots & 0 & \mathbf{I}_n \end{pmatrix} \tag{5.3}$$

où $\mathbf{1}$ est la matrice identité 3×3. Cette matrice $\mathbf{M}$ est diagonale par blocs et symétrique défine positive (SDP).

Notons maintenant $\mathbf{a}_u$ l'accélération (généralisée) non contrainte du groupe de contacts, c'est-à-dire l'empilement des accélérations qu'auraient les objets composant le groupe de contacts s'ils n'étaient pas soumis aux contraintes géométriques. Par exemple, l'accélération non contrainte d'un cube posé sur une table et soumis à la gravité est précisément la gravité. Le vecteur $\mathbf{a}_u$ est lui aussi dans $\mathbb{R}^{6n}$.

Nous pouvons maintenant exprimer le principe de Gauss :

Théorème (Principe des moindres contraintes de Gauss). *A chaque instant, l'accélération généralisée* contrainte $\mathbf{a}_c$ *d'un groupe de contacts minimise la fonction scalaire de* $\mathbf{a}$ *suivante :*

$$G_a(\mathbf{a}) = \frac{1}{2}(\mathbf{a}-\mathbf{a}_u)^T\mathbf{M}(\mathbf{a}-\mathbf{a}_u) = \frac{1}{2}||\mathbf{a}-\mathbf{a}_u||^2_M \tag{5.4}$$

sur l'ensemble des accélérations possibles*, c'est-à-dire celles qui sont compatibles avec la configuration courante du groupe de contacts.*

Remarquons que, puisque la matrice $\mathbf{M}$ est SDP, $||.||_M$ est une norme non-euclidienne bien définie. Par analogie avec l'énergie cinétique d'un système

$$E = \frac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v}$$

cette norme est habituellement appelée *norme cinétique.*

Ainsi, appliquer le principe de Gauss revient à minimiser la distance cinétique entre les accélérations généralisées $\mathbf{a}$ et $\mathbf{a}_u$, sur l'ensemble des accélérations possibles[4].

En rendant implicite le fait que la distance est la distance cinétique, le principe de Gauss peut être formulé encore plus simplement : *à chaque instant, l'accélération contrainte d'un groupe de contact est l'accélération la plus proche possible de son accélération non contrainte.*

Dans cette formulation, les inconnues *explicites* sont les accélérations des objets, et les forces de contact ne sont qu'implicites. Ainsi, le problème est formulé dans *l'espace*

[4]Puisque G_a et $\sqrt{G_a}$ sont minimisées par le même $\mathbf{a}_c$.

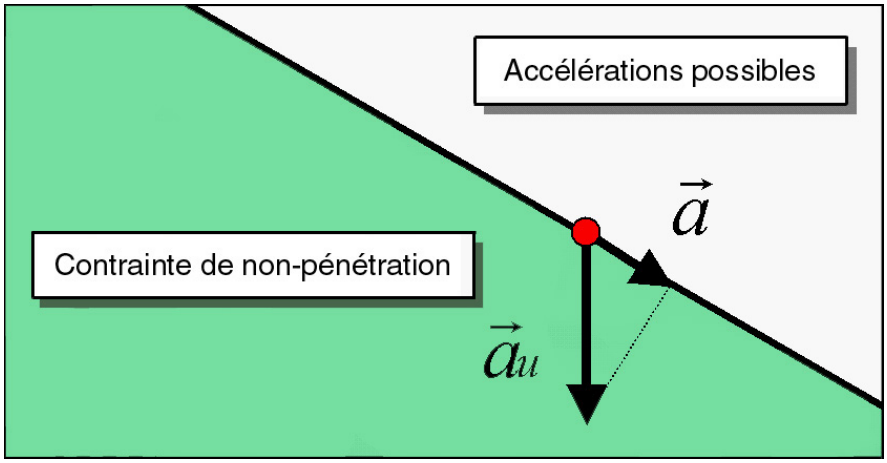


Fig. 5.2 – Selon le principe des moindres contraintes de Gauss, l'accélération contrainte de la particule est la plus proche possible de son accélération non contrainte, la gravité.

des mouvements, et non plus dans l'espace des contacts, comme dans les méthodes LCP [RK97]. Aussi, le nombre de degrés de liberté du groupe de contacts non contraint est maintenant explicite.

La figure 5.2 présente une application du principe de Gauss à une particule glissant sur une pente et soumise à la gravité (l'avantage des exemples ne faisant intervenir qu'une particule est qu'alors l'espace de l'objet et l'espace de ses mouvements peuvent être superposés, et que la distance cinétique est proportionnelle à la distance euclidienne). Dans cet exemple, le principe de Gauss permet de trouver très simplement l'accélération contrainte de la particule. Les accélérations possibles sont celles qui respectent la *contrainte de non-pénétration*, *i.e.* celles dont la composante normale est positive ou nulle. L'accélération contrainte de la particule est la plus proche possible de son accelération non contrainte, la gravité.

5.3 Formulation des problèmes dynamiques

5.3.1 Le problème du mouvement contraint

Le problème du mouvement contraint pour un groupe de contacts de n objets est une application directe du principe de Gauss dans l'espace des accélérations. Il ne reste qu'à exprimer l'ensemble des accélérations possibles.

Cet ensemble peut être déduit du modèle de contact traditionnel [Bar89, Bar94, RK97]. Notons i et j deux objets en contact. I est un point de contact, $\mathbf{n}$ est la normale au plan tangent de contact en I, dirigée de j vers i. Selon l'objet auquel il appartient, le point I est noté I_i ou I_j.

Notons que cette distinction est nécessaire pour établir les équations des contraintes. Cependant, bien que ces deux points coïncident en théorie, ce n'est pas le cas en pratique en raison des imprécisions numériques dues à la précision finie d'un ordinateur d'une part. De plus, nous verrons dans le chapitre suivant que la méthode utilisée pour coupler les algorithmes de détection de collisions aux algorithmes de calcul de mouvement contraint d'autre part impose de séparer les points I_i et I_j qui sont alors, dans la pratique, les points les plus proches appartenant aux primitives polyédriques. Dans le cas d'un contact point/face, par exemple, I_i est le point et I_j est le point de la face le plus proche du point I_i.

Avec cette notation, la *contrainte de non-pénétration* sur les accélérations normales relatives au point de contact I est [Bar89] :

$$(\mathbf{a}_i(I_i) - \mathbf{a}_j(I_j)).\mathbf{n} + 2.(\mathbf{v}_i(I_i) - \mathbf{v}_j(I_j)).\frac{d\mathbf{n}}{dt} \geqslant 0 \tag{5.5}$$

Rappelons que lors du calcul des accélérations contraintes des objets, leurs positions et vitesses sont connues. Aussi le terme $2.(\mathbf{v}_i(I_i) - \mathbf{v}_j(I_j)).\frac{d\mathbf{n}}{dt}$ est-il une constante connue.

Dans l'espace des contacts, les contraintes de ce type seraient utilisées pour former la matrice $\mathbf{A}$ et le vecteur $\mathbf{b}$ donnés en introduction de ce chapitre, qui relient les forces de contact et les accélérations normales *aux points de contact*. Cependant, pour appliquer le principe de Gauss, nous devons exprimer cette contrainte sur les accélérations *des objets*. Ceci est réalisé simplement : pour tout objet k, on sait que

$$\mathbf{a}_k(I_k) = \mathbf{a}_k(G_k) + \alpha_k \wedge \mathbf{G}_k\mathbf{I}_k + \omega_k \wedge (\omega_k \wedge \mathbf{G}_k\mathbf{I}_k) \tag{5.6}$$

où ω_k est la vitesse rotationnelle (connue) de l'objet k.

En substituant cette dernière expression dans l'inégalite (5.5), on obtient une nouvelle contrainte sur les accélérations des objets :

$$(\mathbf{a}_i(G_i) + \alpha_i \wedge \mathbf{G}_i\mathbf{I}_i - \mathbf{a}_j(G_j) - \alpha_j \wedge \mathbf{G}_j\mathbf{I}_j).\mathbf{n} \geqslant K \tag{5.7}$$

où K est une valeur connue, dépendant des vitesses des objets.

En notant que, pour tout triplet $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ de vecteurs de $\mathbb{R}^3$, on a $(\mathbf{a} \wedge \mathbf{b}).\mathbf{c} = \mathbf{a}.(\mathbf{b} \wedge \mathbf{c})$, on montre que la relation (5.7) est bien linéaire en les accélérations des objets en contact :

$$\mathbf{a}_i(G_i).\mathbf{n} + \alpha_i.(\mathbf{G}_i\mathbf{I}_i \wedge \mathbf{n}) - \mathbf{a}_j(G_j).\mathbf{n} - \alpha_j.(\mathbf{G}_j\mathbf{j}_j \wedge \mathbf{n}) \geqslant K \tag{5.8}$$

En empilant les m contraintes de non-pénétration (5.8), on obtient une seule contrainte sur l'accélération généralisée du groupe de contact :

$$\mathbf{Ja} \geqslant \mathbf{c} \tag{5.9}$$

où $\mathbf{J}$ est la jacobienne $m \times 6n$ bien connue du groupe de contacts [RK97], et $\mathbf{c}$ est dans $\mathbb{R}^m$. Cette contrainte généralisée[5] définit l'ensemble convexe des accélérations possibles :

$$\{\mathbf{a} : \mathbf{Ja} \geqslant \mathbf{c}\} \tag{5.10}$$

Ainsi, l'accélération contrainte du groupe de contacts est :

$$\mathbf{a}_c = \arg\min \left\{ \frac{1}{2} ||\mathbf{a} - \mathbf{a}_u||^2_M : \mathbf{Ja} \geqslant \mathbf{c} \right\} \tag{5.11}$$

Dans cette formulation, une propriété bien connue des systèmes d'objets rigides sans frottements est immédiatement visible. Puisque l'accélération contrainte $\mathbf{a}_c$ minimise une distance (non-euclidienne) sur un ensemble convexe, elle est *unique*.

Enfin, remarquons que l'équation (5.11) fournit l'accélération contrainte d'un groupe de contacts pour un nombre quelconque d'objets, soumis à un nombre quelconque de contraintes unilatérales ou bilatérales[6].

5.3.2 Le problème de la réponse à la collision

Montrons maintenant comment formuler le problème de la réponse à la collision dans l'espace des *vitesses* grâce au principe de Gauss.

Notons $\mathbf{v}$ une vitesse généralisée d'un groupe de contacts de n objets. Le vecteur $\mathbf{v}$ est dans $\mathbb{R}^{6n}$. Notons $\mathbf{v}^-$ la vitesse (généralisée) du groupe de contacts immédiatement *avant* une collision. Le problème est de trouver $\mathbf{v}^+$, la vitesse (généralisée) du groupe de contact immédiatement *après* la collision[7]. Pour cela, nous faisons deux hypothèses classiques [RK97] :

- la durée de la collision dt est infinitésimale [MW88],
- la réaction des objets suit le modèle de Newton [MC95].

La conséquence de la première hypothèse est que nous pouvons considérer que les positions des objets, ainsi que les contraintes agissant sur eux, sont constantes durant la période de contact. De plus, les forces extérieures deviennent négligeables face à l'intensité des forces de contact. Par conséquent, l'accélération non contrainte de groupe de contacts est nulle : $\mathbf{a}_u = 0$. L'accélération généralisée $\mathbf{a}$, quant à elle, est infinie. L'impulsion $\mathbf{a}dt$, finie, provoque le changement de vitesse des objets :

$$\mathbf{v}^+ - \mathbf{v}^- = \mathbf{a}dt \tag{5.12}$$

[5] Rappelons qu'une telle inégalité vectorielle s'exerce, par définition, coordonnée par coordonnée.

[6] Puisque toute contrainte bilatérale peut être remplacée par une paire de contraintes unilatérales opposées.

[7] Rappelons que, à chaque fois que l'on calcule des accélérations contraintes, on suppose que les *vitesses* des objets respectent les contraintes. Ce n'est pas le cas lorsque une collision survient : les vitesses des objets immédiatement avant le choc sont incompatibles avec la contrainte de non-pénétration.

Puisque $\mathbf{a}_u = 0$, le principe de Gauss permet d'affirmer que $\mathbf{v}^+$ minimise G_v, où

$$G_v(\mathbf{v}) = \frac{1}{2}||\mathbf{v} - \mathbf{v}^-||^2_M \tag{5.13}$$

sur *l'ensemble des vitesses possibles* immédiatement après la collision.

Cet ensemble des vitesses possibles est trouvé simplement grâce à la seconde hypothèse. Dans le cas de collisions (éventuellement) simultanées, l'hypothèse de Newton fournit la *contrainte de réponse à la collision* sur les vitesses des objets, en chacun des m points de contact (qu'il y ait collision ou non au point de contact) :

$$(\mathbf{v}_i^+(I_i) - \mathbf{v}_j^+(I_j)).\mathbf{n} \geqslant -e(\mathbf{v}_i^-(I_i) - \mathbf{v}_j^-(I_j)).\mathbf{n} \tag{5.14}$$

ou e est le coefficient de restitution au point de contact.

Or, comme précédemment, les vitesses aux points de contact dépendent des vitesses translationnelles et rotationnelles des objets. Pour tout objet k, on sait que

$$\mathbf{v}_k(I_k) = \mathbf{v}_k(G_k) + \omega_k \wedge \mathbf{G}_k\mathbf{I}_k \tag{5.15}$$

où $\mathbf{v}_k(G_k)$ et ω_k sont respectivement la vitesse translationnelle et la vitesse rotationnelle de l'objet k. En remplaçant $\mathbf{v}_i(I_i)$ et $\mathbf{v}_j(I_j)$ dans la contrainte (5.14) par leur expression (5.15), et en notant K le terme de droite (connu) de la contrainte (5.14), on obtient une nouvelle contrainte sur les vitesses translationnelles et rotationnelles des objets i et j :

$$(\mathbf{v}_i^+(G_i) + \omega_i^+ \wedge \mathbf{G}_i\mathbf{I}_i - \mathbf{v}_j^+(G_j) + \omega_j^- \wedge \mathbf{G}_j\mathbf{I}_j).\mathbf{n} \geqslant K \tag{5.16}$$

En utilisant comme précédemment la formule reliant le produit scalaire au produit vectoriel, on montre que cette contrainte est linéaire en les vitesses translationnelles et rotationnelles des objets i et j :

$$\mathbf{v}_i^+(G_i).\mathbf{n} + \omega_i^+.(\mathbf{G}_i\mathbf{I}_i \wedge \mathbf{n}) - \mathbf{v}_j^+(G_j).\mathbf{n} - \omega_j^+.(\mathbf{G}_j\mathbf{j}_j \wedge \mathbf{n}) \geqslant K \tag{5.17}$$

En empilant les m contraintes de réponse à la collision, on obtient l'ensemble des vitesses possibles après collision :

$$\{\mathbf{v} : \mathbf{J}\mathbf{v} \geqslant \mathbf{d}\} \tag{5.18}$$

où $\mathbf{J}$ est une jacobienne $m \times 6n$, semblable à celle de la contrainte sur les accélérations (5.9), et $\mathbf{d}$ est un vecteur de $\mathbb{R}^m$.

Ainsi, la vitesse généralisée du groupe de contacts immédiatement après la collision est :

$$\mathbf{v}^+ = \arg\min \left\{ \frac{1}{2}||\mathbf{v} - \mathbf{v}^-||^2_M : \mathbf{J}\mathbf{v} \geqslant \mathbf{d} \right\} \tag{5.19}$$

Ici aussi le résultat peut s'énoncer d'une façon très élégante : *lors d'une collision, la vitesse d'un groupe de contact immédiatement après la collision est la plus proche possible de la vitesse qu'il avait immédiatement avant la collision.*

La figure 5.3 montre une application de cette loi à une particule qui vient d'entrer en contact avec un objet immobile.

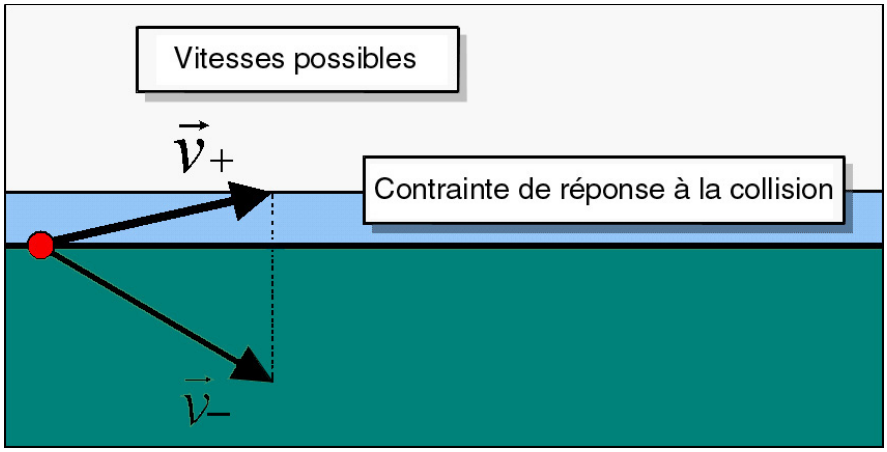


Fig. 5.3 – *Une particule vient de rentrer en contact avec le sol.* Les vitesses possibles sont données par la *contrainte de réponse à la collision*, qui dépend de e, le coefficient de restitution. La vitesse de la particule immédiatement après la collision est la plus proche possible de celle qu'elle avait immédiatement avant la collision.

5.4 Résolution des problèmes dynamiques

La section précédente a réduit les deux problèmes dynamiques au même problème de minimisation dans un espace de mouvements (*i.e.* vitesses ou accélérations) :

$$\mathbf{x}^* = \arg\min \left\{ \frac{1}{2} ||\mathbf{x} - \mathbf{x}_u||_M^2 : \mathbf{J}\mathbf{x} \geqslant \mathbf{c} \right\} \tag{5.20}$$

où $\mathbf{x}_u$, $\mathbf{M}$, $\mathbf{J}$ et $\mathbf{c}$ sont connus. Plus précisément, si le groupe de contacts contient n objets soumis à m contraintes unilatérales, alors $\mathbf{x}$ et $\mathbf{x}_u$ sont dans $\mathbb{R}^{6n}$, $\mathbf{c}$ est dans $\mathbb{R}^m$, et $\mathbf{J}$ est dans $\mathbb{R}^{m \times 6n}$.

Commençons par remarquer que, puisque la matrice $\mathbf{M}$ est symétrique définie positive, elle peut être factorisée en le produit de deux matrices définies positives $6n \times 6n$:

$$\mathbf{M} = \mathbf{Q}^T \mathbf{Q} \tag{5.21}$$

Pour cela, il suffit de diagonaliser $\mathbf{M}$ par blocs (puisque les tenseurs d'inertie $\mathbf{I}_i$ sont des matrices SDP, et que les matrices $m_i\mathbf{1}$ sont déjà diagonales). Il est d'ailleurs utile de noter que ces tenseurs d'inertie peuvent être exprimés en fonction de leurs équivalents *locaux* $\mathbf{I}_i^l$ (*i.e.* évalués dans le repère local de l'objet) :

$$\mathbf{I}_i = \mathbf{P}_i^T \mathbf{I}_i^l \mathbf{P}_i \tag{5.22}$$

où $\mathbf{P}_i$ est la matrice décrivant l'orientation de l'objet i. Comme la factorisation des tenseurs d'inertie locaux peut être précalculée, celle de de $\mathbf{M}$ peut être effectuée en O(n) au cours d'une simulation.

Il est maintenant possible de rendre visibles les forces de contact (ou les impulsions), grâce à la méthode du lagrangien. Cependant, nous *aboutirons à une formulation dans l'espace des mouvements*. Le lagrangien associé au problème (5.20) est :

$$L(\mathbf{x}, \lambda) = \frac{1}{2}||\mathbf{x} - \mathbf{x}_u||^2_M - \lambda^T(\mathbf{J}\mathbf{x} - \mathbf{c}) \tag{5.23}$$

Le vecteur λ est dans $\mathbb{R}^m$ et représente les forces de contact ou les impulsions, selon le problème dynamique.

Puisque

$$\frac{\partial L}{\partial \mathbf{x}} = 0 \Leftrightarrow \mathbf{x} = \mathbf{M}^{-1}\mathbf{J}^T\lambda + \mathbf{x}_u \tag{5.24}$$

la variable $\mathbf{x}$ peut être éliminée (en perdant donc de vue, temporairement, le mouvement des objets).

Ce faisant, nous devons maintenant minimiser :

$$f(\lambda) = \frac{1}{2}\lambda^T\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\lambda - \lambda^T(\mathbf{c} - \mathbf{J}\mathbf{x}_u) \tag{5.25}$$

sous la contrainte de positivité $\lambda \geqslant \mathbf{0}$.

En supposant que l'état du groupe de contact est compatible avec les contraintes, il existe un vecteur $\mathbf{k}$ tel que :

$$\mathbf{J}\mathbf{k} = \mathbf{c} \tag{5.26}$$

Enfin, en utilisant la factorisation de la matrice $\mathbf{M}$, nous pouvons retrouver une formulation dans l'espace des mouvements. Cependant, la distance impliquée est désormais euclidienne. Notons $\mathbf{s} = \mathbf{Q}(\mathbf{k} - \mathbf{x}_u)$ et $\mathbf{J}_Q = \mathbf{J}\mathbf{Q}^{-1} = \mathbf{J}\mathbf{Q}^T$. Minimiser (5.25) est équivalent au problème de moindres carrés non-négatifs suivant :

$$\lambda^* = \arg\min\left\{\frac{1}{2}||\mathbf{J}_Q^T\lambda - \mathbf{s}||^2 : \lambda \geqslant \mathbf{0}\right\} \tag{5.27}$$

Puisque $\mathbf{J}_Q^T\lambda$ et $\mathbf{s}$ sont tous deux dans $\mathbb{R}^{6n}$ et puisque, d'après l'équation (5.24), on a

$$\mathbf{x} = \mathbf{Q}^{-1}\mathbf{J}_Q^T\lambda + \mathbf{x}_u \tag{5.28}$$

cette formulation est bien établie dans l'espace des mouvements.

Géométriquement, le problème (5.27) consiste à projeter le point $\mathbf{s}$ sur le *cône positif* $\mathcal{C}$ engendré par les lignes de la matrice $\mathbf{J}_Q$:

$$\mathcal{C} = \{\mathbf{x} \in \mathbb{R}^{6n} : \mathbf{x} = \mathbf{J}_Q^T\lambda \text{ et } \lambda \geqslant \mathbf{0}\} \tag{5.29}$$

Le problème (5.27) est donc un problème de calcul de point le plus proche.

5.5 Comparaison des formulations

5.5.1 Equivalence mathématique

La formulation équivalente dans l'espace des contacts est obtenue directement à partir du problème (5.27). En effet, les conditions nécessaires et suffisantes (conditions de Karush, Kuhn et Tucker) pour le problème (5.27), sont :

$$\begin{cases} \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\lambda + \mathbf{J}\mathbf{x}_u - \mathbf{c} \geqslant \mathbf{0}, \\ \lambda \geqslant \mathbf{0}, \\ (\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\lambda + \mathbf{J}\mathbf{x}_u - \mathbf{c})^T\lambda = 0 \end{cases} \tag{5.30}$$

Cette formulation est exactement[8] celle donnée dans Baraff [Bar94] ou Ruspini *et al.* [RK97]. En effet, pour obtenir le problème de complémentarité linéaire (5.1) donné en introduction de ce chapitre, il suffit de poser :

$$\begin{cases} \mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T \\ \mathbf{b} = \mathbf{J}\mathbf{x}_u - \mathbf{c} \\ \mathbf{f} = \lambda \end{cases} \tag{5.31}$$

Dans cette formulation, le nombre de degrés de liberté du groupe de contacts est implicite, puisque tous les vecteurs sont dans $\mathbb{R}^m$, et puisque la factorisation de la matrice d'inertie opérationnelle $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ est cachée.

A ce point du raisonnement, le lecteur pourrait se demander pourquoi nous n'avons pas directement dérivé le problème (5.27) de la formulation LCP (5.30). Après tout, il suffit de factoriser la matrice des masses $\mathbf{M}$ et de remarquer que $\mathbf{J}\mathbf{x}_u - \mathbf{c}$ est dans l'espace image de $\mathbf{J}\mathbf{Q}^{-1}$. Cependant, nous pensons que l'exploration des principes physiques intuitifs offerts par le principe de Gauss aidera à comprendre pourquoi un algorithme résolvant le problème (5.27) devrait être plus efficace qu'un autre résolvant le problème (5.30).

5.5.2 Différence algorithmique

Bien que les problèmes (5.27) et (5.30) soient mathématiquement équivalents, ils ne le sont pas d'un point de vue *algorithmique*, pour essentiellement quatre raisons :

[8] Notons toutefois que, par souci de clarté, les formulations données dans ce chapitre concernent des objets rigides classiques à six degrés de liberté. Cependant, le principe de Gauss reste valide pour des systèmes mécaniques articulés à nombre de degrés de liberté quelconque (ces systèmes, décrits par un ensemble de coordonnées réduites, internalisent simplement les contraintes bilatérales), et peut donc être utilisé avec le modèle de contact décrit dans Ruspini [RK97]. Par ailleurs, l'équivalence mathématique donnée dans cette section en est une preuve, puisqu'il suffit de factoriser la matrice $\mathbf{M}$.

- *meilleur conditionnement* : le conditionnement de la matrice $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ est égal au carré du conditionnement de la matrice $\mathbf{J}_Q$ (voir Gill *et al.* [GMW81], par exemple). En conséquence, le problème formulé dans l'espace des mouvements (5.27) est beaucoup mieux conditionné que le problème formulé dans l'espace des mouvements (5.30). Il est donc beaucoup moins sensible aux erreurs d'arrondi ;
- *formulation creuse* : Puisque les matrices $\mathbf{M}$ et $\mathbf{Q}$ sont diagonales par blocs, et que la jacobienne $\mathbf{J}$ est toujours creuse, la matrice $\mathbf{J}_Q$ intervenant dans le problème (5.27) est elle aussi *toujours creuse*. Au contraire, la matrice $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ peut être dense [Bar96] ;
- *meilleur encombrement mémoire* : Lorsque les matrices impliquées sont traitées comme des matrices denses, il faut beaucoup plus de mémoire pour stocker $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ $(O(m^2))$ que pour stocker $\mathbf{J}_Q$ $(O(nm))$, lorsque le nombre moyen de points de contact par objet augmente. De plus, lorsqu'elles sont traitées comme des matrices creuses, l'encombrement mémoire de $\mathbf{J}_Q$ est *toujours* en $O(m)$ seulement, alors que $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ peut être dense ;
- *pas de calculs redondants* : un algorithme qui résout le problème LCP doit faire respecter les conditions à la fois sur les forces de contact (ou les impulsions) λ et les accélérations (ou les vitesses) aux points de contact $\mathbf{x}_{cp}$. En conséquence, cet algorithme doit maintenir les m coordonnées de $\mathbf{x}_{cp}$ pendant la résolution du problème (5.30). Cependant, les m coordonnées de $\mathbf{x}_{cp}$ ne sont pas indépendantes, puisque les points de contacts appartiennent à des objets rigides : ces coordonnées pourraient être déduites des mouvements des objets. Puisque les mouvements des objets ne sont pas immédiatement disponibles dans l'espace des contacts, cet algorithme effectue des calculs redondants, donc non nécessaires. Dans l'espace des mouvements, un algorithme peut opérer directement sur les $6n$ coordonnées indépendantes seulement, et sur les coefficients correspondants dans $\mathbf{J}_Q$.

En guise de conclusion préliminaire, nous pouvons exprimer plus simplement la différence entre les deux formulations : dans l'espace des contacts, les forces de contact sont calculées *pour elles-mêmes*, et sont *ensuite* utilisées pour calculer le mouvement des objets. Dans l'espace des mouvements, les forces de contact sont calculées *en même temps que les mouvements des objets*[9].

5.5.3 Comparaison expérimentale

Il existe de nombreuses façons de résoudre le problème de point le plus proche (5.30). Cependant, sa structure géométrique a conduit les chercheurs en optimisation à développer des algorithmes spécifiques, généralement répartis en deux catégories : les méthodes *combinatoires* (méthodes d'ensemble actif), et les méthodes de *descente*. Les algorithmes de la première catégorie cherchent la solution en se déplaçant de face en face sur le cône positif. Ceux de la seconde catégorie sont des méthodes de point intérieur ou

[9] Puisque les mouvements des objets sont directement reliés à $\mathbf{J}_Q^T\lambda$.

de point extérieur [AS90]. Pour des simulations dynamiques d'objets rigides typiques, cependant, il est plus simple d'utiliser les méthodes combinatoires. C'est d'ailleurs le cas dans les formulations situées dans l'espace des contacts : au moins pour les systèmes sans frottements, les problèmes LCP sont résolus par l'algorithme de Lemke ou de Dantzig [APS99, Bar94, RK97, SS98a, TPSL95].

Afin de comparer expérimentalement la formulation dans l'espace des contacts à celle dans l'espace des mouvements, nous avons choisi d'implanter l'algorithme bien connu de Baraff [Bar94] pour résoudre le problème (5.30). Dans le cas de systèmes sans frottements, cet algorithme est équivalent à l'algorithme de Dantzig qui résout les problèmes de complémentarité linéaire. Pour le problème de point le plus proche (5.27), nous avons choisi l'algorithme de Wilhelmsen [Wil76] pour plusieurs raisons. C'est une méthode combinatoire, facile à implémenter et qui, surtout, présente un rapport étroit avec l'algorithme de Dantzig, rendant ainsi la comparaison plus pertinente. Ainsi, comme la plupart des méthodes par pivot, les deux algorithmes ont une boucle interne et une boucle externe. Dans la boucle interne, les deux algorithmes doivent résoudre des systèmes linéaires impliquant des sous-matrices de $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ (remarquons que, par conséquent, l'argument concernant le meilleur conditionnement des méthodes dans l'espace des mouvements ne tient plus. Cependant, ceci permet de comparer les deux algorithmes sous des hypothèses de robustesse équivalentes). Ainsi, la différence entre les deux méthodes réside dans le fait que l'algorithme de Wilhelmsen opère sur $\mathbf{J}_Q^T\lambda$ et λ, alors que celui de Dantzig opère sur $\mathbf{x}_{cp}$ et λ.

	n=2	n=7	n=12	n=17	n=22
p=5	0.60	0.50	0.51	0.54	0.52
p=15	0.85	1.08	1.23	1.47	1.63
p=25	0.85	0.90	1.07	1.21	1.05
p=35	0.94	0.93	1.31	1.15	1.01
p=45	1.01	1.08	1.5	1.08	1.03

Tab. 5.1 – Evolution du rapport des temps d'exécution en fonction de n, le nombre d'objets, et p, le nombre de points de contact par objets. Dans le cas dense, le rapport des temps d'exécution ne présente pas de comportement significatif.

Nous avons utilisé la même ligne de conduite pour implanter les deux algorithmes. La matrice $\mathbf{J}_Q$ et la matrice $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ étaient considérées comme denses dans une version des algorithmes, et comme creuse dans une autre. Une factorisation de Cholesky était utilisée pour résoudre les systèmes linéaires dans tous les cas. Cependant, nous n'avons pas implanté une méthode de factorisation incrémentale, pour aucun des algorithmes, comme suggéré dans [Bar94]. Comme des tests préliminaires avaient montré que les deux algorithmes effectuaient approximativement le même nombre de résolutions de systèmes linéaires, aucun algorithme n'en aurait bénéficié significativement plus que l'autre. Aussi, les deux algorithmes utilisaient la même valeur de tolérance : $\epsilon = 10^{-5}$.

Notons que nous sommes tout à fait conscients que, malgré tous nos efforts pour

implanter les algorithmes de la même façon, il peut nous être reproché qu'il n'est pas toujours pertinent de comparer des temps d'exécution. Cependant, nous sommes beaucoup moins intéressés par le rapport des temps d'exécution que par le comportement de ce rapport.

	n=2	n=7	n=12	n=17	n=22
p=5	0.51	0.62	0.80	0.91	0.93
p=15	0.71	1.19	1.52	1.95	2.19
p=25	0.70	1.04	1.60	1.98	1.90
p=35	0.76	1.11	2.08	2.20	2.05
p=45	0.78	1.34	2.44	2.43	2.20

Tab. 5.2 – Evolution du rapport des temps d'exécution en fonction de n, le nombre d'objets, et p, le nombre de points de contact par objets. Dans le cas creux, la formulation dans l'espace des mouvements tire parti de la formulation creuse et est plus efficace à mesure que le nombre de points de contact par objet augmente.

Les tests se sont déroulés de la manière suivante. Pour un test donné, une matrice aléatoire $\mathbf{J}$ était calculée ainsi : pour chaque rangée, deux entiers étaient sélectionnés aléatoirement entre 1 et n (les indices des objets). Ces deux entiers (éventuellement égaux, la contrainte représentant alors un contact avec un objet immobile du décor), étaient utilisés pour placer deux ensembles de six coefficients dans la rangée, conduisant ainsi à une jacobienne typique. Pour obtenir un vecteur $\mathbf{c}$ compatible avec $\mathbf{J}$, un vecteur aléatoire $\mathbf{k}$ était choisi dans $\mathbb{R}^{6n}$ pour calculer $\mathbf{c} = \mathbf{J}\mathbf{k}$. Les données du problème de complémentarité linéaire équivalent étaient calculées comme dans l'équation (5.30), en prenant pour $\mathbf{M}$ la matrice identité $6n \times 6n$ (ce qui n'avait évidemment pas de conséquence sur la comparaison). Les deux paramètres des tests étaient le nombre d'objets n et le nombre (moyen) de contraintes unilatérales par objet p. Pour une paire donnée (n, p), cinquante temps d'exécution étaient obtenus pour les algorithmes de Dantzig et de Wilhelmsen : t_i^D et t_i^W, respectivement, pour i variant de 1 à 50. Le rapport des temps d'exécution était donc :

$$r(n,p) = \frac{\sum_{i=1}^{50} t_i^D}{\sum_{i=1}^{50} t_i^W} \tag{5.32}$$

Remarquons que les entiers n et p étaient choisis de façon que les deux problèmes puissent tenir en mémoire et ne requièrent pas d'accès disque.

Les résultats sont rapportés dans le tableau 5.1 pour le cas dense (*i.e.* dense pour les deux algorithmes), et dans le tableau 5.2 pour le cas creux (*i.e.* creux pour les deux algorithmes). Insistons à nouveau sur le fait que nous ne sommes pas vraiment intéressés par le rapport d'exécution lui-même, mais par son comportement, visible dans les tableaux 5.1 et 5.2 : l'algorithme opérant dans l'espace des mouvements tire parti de la formulation creuse pour être de plus en plus efficace, par rapport à celui opérant dans l'espace des contacts, à mesure que le nombre moyen de contraintes

par objet augmente. Si l'on considère maintenant le rapport d'exécution lui-même, et en considérant le fait que les mêmes règles ont été appliquées pour implanter les deux algorithmes, nous pouvons (prudemment) affirmer qu'un algorithme opérant dans l'espace des mouvements est plus efficace qu'un autre opérant dans l'espace des contacts (au moins pour notre implantation et pour les tests que nous avons effectués).

Il devrait être clair cependant que, lorsque les objets sont simples et/ou qu'il y a peu de contraintes par objet, un algorithme travaillant dans l'espace des contacts devrait (comme l'on peut s'y attendre) être plus efficace. Lorsque la complexité des objets augmente, cependant, la formulation dérivée du principe de Gauss semble préférable.

5.6 Un modèle de friction dans l'espace des mouvements

Le principe de Gauss n'est valable que pour les systèmes sans frottements. Cependant, l'intérêt de développer un modèle de friction dans l'espace des mouvement est de conserver les bonnes propriétés mathématiques obtenues pour les systèmes sans frottements. Nous présentons maintenant un modèle de friction, empirique, inspiré du modèle de Coulomb, permettant de simuler la friction statique et la friction dynamique. Bien entendu, en raison du caractère tout à fait empirique de notre modèle (différent, bien qu'il en soit inspiré, du modèle de Coulomb), nous ne prétendons pas qu'il convient à des simulations physiques rigoureuses (alors que le principe de Gauss décrit précédemment permet de décrire exactement l'évolution au cours du temps des systèmes sans frottements). Cependant, les simulations effectuées permettent de constater que ce modèle est satisfaisant du point de vue d'un observateur humain, et laissent penser qu'il peut être utilisé au moins dans des applications infographiques ou de réalité virtuelle.

Notons enfin, avant de détailler le modèle proposé, que nous aurions pu tenter de modifier un algorithme résolvant les problèmes dynamiques dans l'espace des mouvements (l'algorithme de Wilhelmsen par exemple), afin d'intégrer le modèle de frottements de Coulomb. Nous avons cependant souhaité conserver l'esprit de la résolution dans l'espace des mouvements, dans lequel les forces de contact ne sont pas directement accessibles, puisqu'elles ne sont que des inconnues implicites utilisées en quelque sorte de façon invisible par l'algorithme de résolution.

5.6.1 Friction dynamique

La friction dynamique s'applique aux points de contacts auxquels la vitesse relative tangentielle (*i.e.* de glissement) est non nulle.

Notons i et j les objets en contact, et notons $\mathbf{n}$ la normale à la surface au point de contact. Cette normale est connue. Considérons dans un premier temps que j est

immobile. L'accélération normale contrainte $\mathbf{a}.\mathbf{n}$ du point de contact I_i, appartenant à l'objet i, peut être exprimée comme la somme de son accélération normale non contrainte $\mathbf{a}_u.\mathbf{n}$ et de l'accélération normale due à la contrainte de non-pénétration $\mathbf{a}_c.\mathbf{n}$. Ainsi :

$$\mathbf{a}.\mathbf{n} = \mathbf{a}_u.\mathbf{n} + \mathbf{a}_c.\mathbf{n} \tag{5.33}$$

Supposons que $\mathbf{v}_t$, la vitesse tangentielle de I_i, soit dirigée suivant le vecteur unitaire $\mathbf{t}$:

$$\mathbf{t} = \mathbf{v}_t / \|\mathbf{v}_t\| \tag{5.34}$$

Nous savons par hypothèse que $\|\mathbf{v}_t\| \neq 0$. Suivant ce vecteur $\mathbf{t}$, l'accélération devrait être $\mathbf{a}_u.\mathbf{t}$. Selon le modèle de Coulomb, cependant, une force de frottement opposée à la vitesse et proportionnelle à la réaction du support est présente : $\mathbf{f}_T = -||\mathbf{f}_N||.\mathbf{t}$. $\mathbf{f}_T$ et $\mathbf{f}_N$ sont respectivement les composantes tangentielles et normales de la force de frottements. Dans le cas simple d'une particule par exemple, cette force de frottement se manifeste par une décélération de la particule, autrement dit par une modification de son accélération contrainte tangentielle.

Aussi, nous proposons de simuler la friction dynamique dans l'espace des mouvements en contraignant directement l'accélération contrainte tangentielle (ce qui ne correspond pas au modèle de Coulomb, qui n'impose rien sur l'accélération contrainte) :

$$\mathbf{a}.\mathbf{t} \leqslant \mathbf{a}_u.\mathbf{t} - \mu\left(\mathbf{a}_c.\mathbf{n}\right) \tag{5.35}$$

où μ est le coefficient de frottement au point de contact. La perte d'accélération contrainte tangentielle est alors $\mu\left(\mathbf{a}_c.\mathbf{n}\right)$. Puisque $\mathbf{a}.\mathbf{n} = \mathbf{a}_u.\mathbf{n} + \mathbf{a}_c.\mathbf{n}$, cette perte est proportionnelle à la perte d'accélération non contrainte normale due à la contrainte de non-pénétration. Ainsi, l'inéquation (5.35) donne :

$$\mathbf{a}.\mathbf{d} \leqslant \mathbf{a}_u.\mathbf{d} \tag{5.36}$$

où $\mathbf{d} = \mathbf{t} + \mu\mathbf{n}$ peut être compris comme un *vecteur de friction dynamique*, dont la direction dépend du coefficient de frottement au point de contact.

Si l'on considère maintenant le cas de deux objets mobiles, les accélérations $\mathbf{a}$ et $\mathbf{a}_u$ sont les accélérations contrainte et non contrainte de I_i *relativement à l'objet* j. Afin d'obtenir la contrainte due à la friction dynamique dans le repère global, ces accélérations doivent être exprimées en fonction des accélérations de I_i et I_j dans le repère global.

Notons $\mathbf{a}_i\left(I_i\right)$ et $\mathbf{a}_{i.u}\left(I_i\right)$ les accélérations de I_i dans le repère global, et notons $\mathbf{a}_j\left(I_j\right)$ et $\mathbf{a}_{j.u}\left(I_j\right)$ les accélérations (contrainte et non contrainte) de I_j dans le repère global. Les formules classiques de mécanique donnent :

$$\mathbf{a} = \mathbf{a}_i\left(I_i\right) - \mathbf{a}_j\left(I_j\right) - 2\omega_j \wedge \left(\mathbf{v}_i\left(I_i\right) - \mathbf{v}_j\left(I_j\right)\right) \tag{5.37}$$

et

$$\mathbf{a}_u = \mathbf{a}_{i.u}\left(I_i\right) - \mathbf{a}_{j.u}\left(I_j\right) - 2\omega_j \wedge \left(\mathbf{v}_i\left(I_i\right) - \mathbf{v}_j\left(I_j\right)\right) \tag{5.38}$$

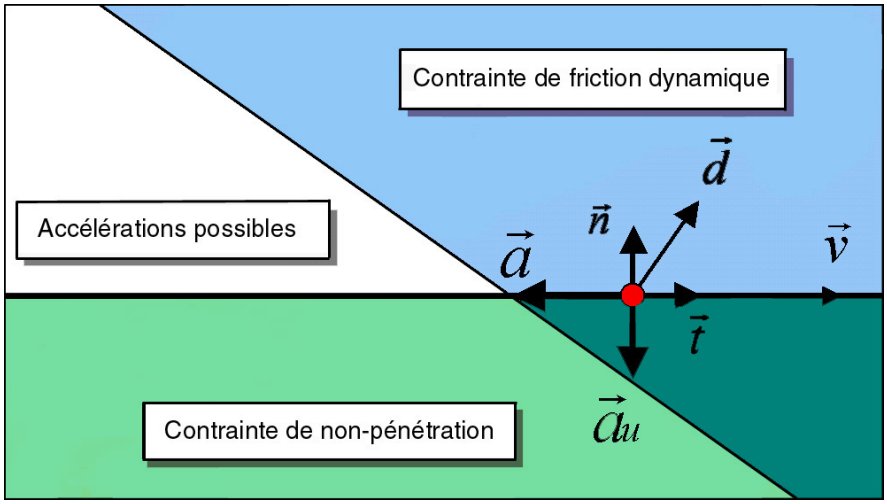


Fig. 5.4 – *Une particule est soumise à la friction dynamique.* La friction dynamique est prise en compte en ajoutant une *contrainte de friction dynamique* à la contrainte de non-pénétration. Cette nouvelle contrainte dépend de l'accélération non contrainte $\mathbf{a}_u$ et du *vecteur de friction dynamique* $\mathbf{d} = \mathbf{t} + \mu\mathbf{n}$, où μ est le coefficient de friction dynamique. L'accélération contrainte de la particule $\mathbf{a}$ est la plus proche possible de son accélération non contrainte.

Ainsi, la *contrainte de friction dynamique* imposée sur les acccélérations des objets est :

$$\left(\mathbf{a}_i\left(I_i\right) - \mathbf{a}_j\left(I_j\right)\right).\mathbf{d} \leqslant \left(\mathbf{a}_{i.u}\left(I_i\right) - \mathbf{a}_{j.u}\left(I_j\right)\right).\mathbf{d} \tag{5.39}$$

où le membre de droite de l'inégalité est connu. Comme précédemment, l'inéquation (5.39) est une fonction linéaire des accélérations translationnelles et rotationnelles des objets i et j.

Par conséquent, dans notre modèle, tout point de contact sujet à du frottement dynamique impose deux contraintes sur les accélérations des objets : la contrainte de non-pénétration et la contrainte de friction dynamique (5.39).

Notons que dans le cas d'une seule particule en contact avec un objet immobile, les inéquations (5.8) et (5.39) utilisées en combinaison avec le principe de Gauss produisent le même mouvement que celui obtenu en appliquant le modèle de Coulomb pour la loi de friction dynamique, comme le montre la figure 5.4.

Cependant, comme dans notre modèle l'ensemble des accélérations possibles reste convexe, le mouvement de l'objet est déterminé de façon unique, ce qui est contraire au modèle de Coulomb. Nous avons vu en effet dans le premier chapitre que le modèle de Coulomb utilisé conjointement avec l'hypothèse de rigidité absolue des objets conduit parfois à des configurations indéterminées, dans lesquelles plusieurs mouvements sont possibles (ce qui est bien sûr contraire à la réalité).

5.6.2 Friction statique

La friction statique s'applique aux points de contact auxquels la vitesse relative tangentielle est nulle. Dans une méthode de simulation par contraintes, la friction statique est prise en compte lors du calcul des accélérations contraintes (le premier problème dynamique) et lors du calcul des vitesses post-impact (le second problème dynamique).

Selon le modèle de Coulomb, le frottement tente d'empêcher la mise en mouvement du point de contact. Tant que la force de frottement vérifie $||\mathbf{f}_T|| \leqslant \mu||\mathbf{f}_N||$, l'accélération tangentielle du point de contact est nulle. Si, par contre, l'accélération tangentielle du point de contact est non nulle, alors la force de frottement est opposée à l'accélération tangentielle, et vérifie $||\mathbf{f}_T|| = \mu||\mathbf{f}_N||$.

Pour simuler la friction statique, nous proposons de (virtuellement) modifier la surface locale. Au lieu de supposer que la surface est localement plane, de normale $\mathbf{n}$, nous imaginons qu'il existe un petit creux pyramidal, dans lequel le point de contact I_i est tombé.

Cette pyramide a un nombre de côtés p arbitrairement fixé, dont les normales $\mathbf{s}_k$ $(1 \leqslant k \leqslant p)$, dirigées vers l'extérieur du creux, dépendent de p et du coefficient de frottement μ. Ainsi, si $(\mathbf{t}_1, \mathbf{t}_2)$ est une base orthonormale de la surface de contact (initialement plane), les p *vecteurs de friction statique* sont :

$$\mathbf{s}_k = \frac{1}{\sqrt{1+\mu^2}} \left(\mathbf{n} - \mu \left(\cos\left(\frac{2k\pi}{p}\right) \mathbf{t}_1 + \sin\left(\frac{2k\pi}{p}\right) \mathbf{t}_2 \right) \right) \tag{5.40}$$

pour $1 \leqslant k \leqslant p$.

Par conséquent, pour simuler la friction statique en un point de contact, p contraintes de non-pénétration (5.8) sont ajoutées à l'ensemble de contraintes, en remplaçant le vecteur $\mathbf{n}$ par le vecteur $\mathbf{s}_k$ dans la k-ième contrainte[10]. La contrainte de non-pénétration initiale n'est plus nécessaire. Au final, tout point de contact soumis à du frottement statique crée p *contraintes de friction statiques*.

Plusieurs remarques peuvent être faites.

– ce modèle empirique correspond intuitivement au comportement d'une particule soumise à du frottement statique, comme le montre la figure 5.5. Tant que l'accélération non contrainte de la particule est suffisamment dirigée vers le bas, la friction statique empêche tout mouvement. Dès que la composante tangentielle de son accélération non contrainte est suffisamment importante, cependant, la particule se met en mouvement[11] ;

[10] De plus, même pour les contacts arête/arête, $d\mathbf{n}/dt$ est remplacé par $\omega_j \wedge \mathbf{s}_k$.

[11] Remarquons que lorsque la particule se met en mouvement, le contact se rompt puisque la composante normale de l'accélération est strictement positive. Ceci ne correspond pas au modèle de Coulomb. Dans notre modèle, par conséquent, il ne peut y avoir de transition roulement/glissement. Cependant, la méthode de couplage décrite dans le chapitre 6, qui impose de maintenir une distance de sécurité entre les objets, rend ces transitions possibles puisqu'un contact n'est rompu que lorsque la distance entre les objets est supérieure à une certaine valeur seuil strictement positive.

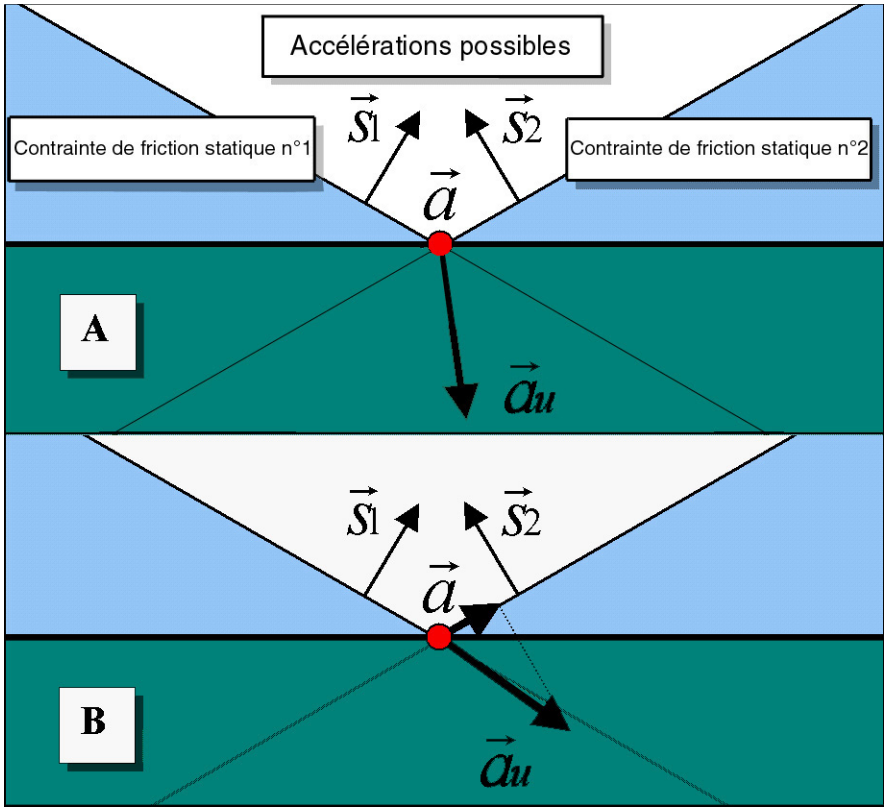


Fig. 5.5 – *Une particule est soumise à la friction statique.* La friction statique est prise en compte en ajoutant un nombre arbitrairement fixé de *contraintes de friction statiques* à la contrainte de non-pénétration. Ces contraintes dépendent de μ, le coefficient de de frottement. De nouveau, l'accélération contrainte de la particule est la plus proche possible de son accélération non contrainte. A : Les frottements statiques empêchent le mouvement. B : Les frottements statiques, trop faibles, ne peuvent empêcher le mouvement.

- comme dans Stewart *et al.* [ST96], le cône de friction a été remplacé par un cône pyramidal, ce qui conduit à du frottement anisotrope. Cependant, ce défaut peut être réduit en augmentant le nombre de côtés de la pyramide p. Grâce aux bonnes propriétés mathématiques de la formulation dans l'espace des mouvements, ceci n'est pas véritablement un problème. Nous avons constaté que fixer p à 6 permet de donner un comportement suffisamment réaliste pour l'oeil humain ;
- puisqu'ici encore le frottement est pris en compte en ajoutant des contraintes linéaires dans l'espace des accélérations, l'ensemble des accélérations possibles reste convexe et le mouvement de l'objet est encore déterminé de manière unique.

Ceci conclut la description d'un modèle de friction simple et empirique. Nous avons bien noté que ce modèle ne correspond pas au modèle de Coulomb, mais nous avons pu *observer* au cours de nos expériences qu'il permet d'obtenir un comportement réaliste.

Notons que ce modèle de frottement peut être appliqué au problème de la réponse à la collision, en transposant les contraintes de friction dynamiques et statiques dans l'espace des vitesses. Pour la friction dynamique, les accélérations sont remplacées par des vitesses dans la contrainte (5.39) (et les accélérations non contraintes sont remplacées par les vitesses des objets immédiatement avant la collision). Pour la friction statique, la contrainte (5.17) est conservée et p contraintes de friction statique dans l'espace des vitesses sont ajoutées :

$$\left(\mathbf{v}_i\left(I_i\right) - \mathbf{v}_j\left(I_j\right)\right).\mathbf{s}_k \geqslant 0 \tag{5.41}$$

pour $1 \leqslant k \leqslant p$.

5.7 Conclusion

Dans ce chapitre, nous avons montré qu'utiliser le principe de Gauss pour formuler les problèmes dynamiques dans l'espace des mouvements permet d'obtenir une formulation mathématiquement équivalente à celle traditionnellement donnée dans l'espace des contacts, mais qui présente plusieurs avantages algorithmiques.

La formulation dans l'espace des mouvements est ainsi mieux conditionnée, toujours creuse, requiert moins de mémoire et évite les calculs redondants. Nous avons pu constater qu'elle permet de tirer parti de la formulation naturellement creuse du problème pour être de plus en plus efficace (relativement à la formulation dans l'espace des contacts), à mesure que le nombre moyen de contacts *par degré de liberté* augmente.

Car, plus que le nombre d'objets, le paramètre important dans la simulation est le nombre de degrés de liberté. Ainsi, une simulation en coordonnées réduites [RK97] bénéficierait encore plus d'une formulation dans l'espace des mouvements.

Nous avons aussi présenté un modèle de friction dans l'espace des mouvements qui, s'il ne correspond pas au modèle de Coulomb, permet d'obtenir un comportement visuellement réaliste.

Dans le chapitre suivant, nous présentons la façon dont les algorithmes de détection de collisions continue introduits dans les chapitres précédents ont été couplés avec les algorithmes de résolution des problèmes dynamiques introduits dans ce chapitre pour former un simulateur dynamique interactif complet, et nous décrivons plusieurs applications de ce simulateur, notamment pour la simulation interactive avec retour d'efforts.

Chapitre 6

Applications et résultats

Les algorithmes de détection de collisions continue et de calcul du mouvement contraint ont été implantés et regroupés dans une librairie C++, CONTACT Toolkit. Ce chapitre présente plusieurs applications et tests de cette librairie. Nous montrons d'une part que les algorithmes introduits dans ce mémoire semblent exploitables dans un contexte industriel et, d'autre part, qu'ils peuvent être utilisés simplement pour la simulation interactive avec retour d'efforts, comme cela a été montré avec diverses configurations haptiques.

6.1 Introduction

Dans les précédents chapitres, nous avons introduit la notion de mouvement intermédiaire arbitraire en détection de collisions continue. Nous avons ainsi présenté deux approches fondé sur cette notion et montré qu'elle permettait de concevoir des algorithmes efficaces pour détecter des collisions entre des modèles polyédriques rigides complexes. Nous avons également montré que le *mouvement* des objets pouvait être exploité pour accélérer la détection de collisions. Par ailleurs, nous avons comparé l'approche traditionnelle pour résoudre les problèmes dynamiques, formulée dans l'espace des contacts, à celle découlant du principe de moindres contraintes de Gauss, formulée dans l'espace des mouvements, et montré que cette dernière présente plusieurs avantages algorithmiques. Ceci nous a conduit à proposer un modèle de friction dans l'espace des mouvements.

Les algorithmes de détection de collisions exposés aux chapitres 2, 3 et 4 ont été regroupés dans une librairie C++ appelée CONTACT (pour *CONTinuous and Accurate collision Tracking*). Cette librairie est portable et a été testée sur des environnements Windows, Unix et Linux.

Les algorithmes de calcul de mouvement contraint, présentés au chapitre 5 ont ensuite été inclus dans CONTACT pour former une nouvelle librairie, CONTACT

Toolkit, testée avec succès sur les mêmes environnements. Le couplage des algorithmes de détection de collisions et de calcul de mouvement contraint s'est réalisé de manière classique [Bar95], en suivant l'algorithme général présenté au chapitre 1.

Un convertisseur d'objets, du format *Inventor* vers le format adapté à CONTACT Toolkit, pouvant contenir la description de l'objet, de sa hiérarchie englobante et de ses cônes de recul, a permis de tester un grand nombre de bases de données et de configurations.

Nous présentons dans ce chapitre plusieurs applications de la librairie, notamment dans certains cas industriels, et montrons que le algorithmes introduits dans ce mémoire peuvent être utilisés simplement pour réaliser une simulation interactive avec retour d'efforts.

La section 6.2 présente la manière de coupler des algorithmes de détection de collisions continue reposant sur des mouvements intermédiaires arbitraires à ceux du calcul de mouvement contraint. La section 6.3 présente plusieurs applications sur station de travail, notamment dans des cas industriels fournis par Renault et Airbus-EADS. La section 6.4 montre comment les algorithmes ont été utilisés pour la simulation interactive avec retour d'efforts, et présente des applications du simulateur sur plusieurs configurations haptiques. Enfin, la section 6.5 conclut le chapitre.

6.2 Couplage des algorithmes de détection de collisions et de calcul du mouvement contraint

L'utilisation d'un mouvement intermédiaire arbitraire pour interpoler les positions successives des objets dans une simulation dynamique par contraintes conduit à maintenir en permanence une petite *distance de sécurité* entre les objets en contact. La plupart du temps, en effet, le mouvement intermédiaire fixé n'est pas compatible avec les contraintes dynamiques au cours des intervalles de temps successifs.

Considérons par exemple le cas d'un parallélépipède en contact continu avec une surface plane, visible dans la figure 6.1.a. Dans cet exemple, le point de contact I devrait être en permanence sur le plan de contact $\mathcal{P}$ au cours de l'intervalle de temps considéré $[t_n, t_{n+1}]$. Cependant, le mouvement réel de l'objet au cours de cet intervalle de temps est remplacé par un mouvement arbitrairement fixé qui, s'il respecte bien les positions de l'objet aux instants t_n et t_{n+1}, ne garantit pas que le point I reste sur le plan $\mathcal{P}$ au cours de l'intervalle de temps. Selon les positions intiales et finales du parallélépipède, qui imposent le mouvement intermédiaire utilisé pour la détection de collisions, ce point I peut en effet, par exemple, souhaiter traverser la surface de contact *dès l'instant* t_n, entrainant la détection d'une collision à l'instant t_n. Afin d'éviter que le simulateur ne reste bloqué à l'instant t_n ou, plus généralement, qu'il ne détecte trop souvent ces collisions artificiellement engendrées par l'utilisation d'un mouvement intermédiaire arbitraire, nous maintenons une distance de sécurité ϵ_s entre les objets

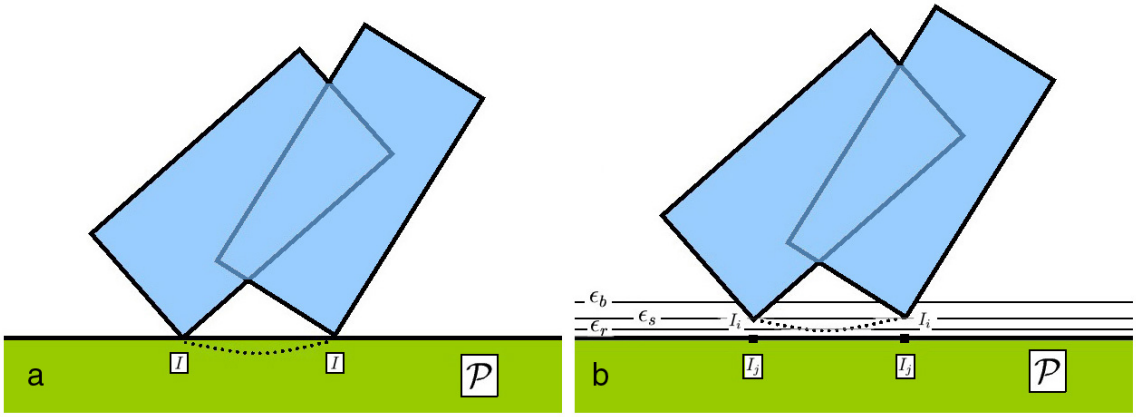


Fig. 6.1 – *a* : le mouvement intermédiaire arbitraire utilisé pour la détection de collisions ne respecte pas les contraintes au cours de l'intervalle de temps $[t_n, t_{n+1}]$. *b* : les objets en contact sont maintenus légèrement éloignés les uns des autres.

en contact, comme représenté dans la figure 6.1.b. Pour cela, les contraintes imposées aux accélérations ou aux vitesses sont légèrement modifiées. Par exemple, *lorsque la distance*[1] *entre les deux objets est inférieure à la distance de sécurité* ϵ_s , la contrainte de non-pénétration (5.8) devient :

$$\mathbf{a}_i(G_i).\mathbf{n} + \alpha_i.(\mathbf{G}_i\mathbf{I}_i \wedge \mathbf{n}) - \mathbf{a}_j(G_j).\mathbf{n} - \alpha_j.(\mathbf{G}_j\mathbf{j}_j \wedge \mathbf{n}) \geqslant K + k(\epsilon_s - d) \tag{6.1}$$

où k est une constante de couplage, et d est la distance du point I_i au point I_j, situés comme indiqué dans la figure 6.1.b. Rappelons en effet que les points I_i et I_j sont les points les plus proches sur les primitives polyédriques.

Deux autres valeurs, ϵ_b et ϵ_r, sont utilisées en combinaison avec la distance de sécurité ϵ_s. La première, ϵ_b, supérieure à ϵ_s, est la valeur au-delà de laquelle le simulateur déclare que le contact entre les deux objets est rompu. La seconde, ϵ_r, inférieure à ϵ_s, est une valeur d'alerte en-deça de laquelle le simulateur arrête le temps de la simulation et effectue un *cycle de repositionnement des objets*, parce qu'il n'est pas parvenu à les maintenir à une distance raisonnable au cours de la simulation. Ce cycle de repositionnements est effectué entre deux affichages successifs de la scène en calculant les plus petits déplacements qui permettent de satisfaire les *contraintes de repositionnement*, semblables aux contraintes de réponse à la collision (5.14) du chapitre précédent :

$$(\mathbf{v}_i(I_i) - \mathbf{v}_j(I_j)).\mathbf{n} \geqslant \epsilon_s - d \tag{6.2}$$

Une fois ces vitesses calculées, elles sont utilisées pour replacer les objets (en détectant des collisions). Tant que certaines contraintes de repositionnement ne sont pas respectées, un nouveau repositionnement est effectué. Nous avons pu observer que les cycles de repositionnement sont généralement très courts, quasiment imperceptibles par l'utilisateur, et n'entravent pas le bon déroulement de la simulation interactive.

[1] Plus précisément, il faut parler de distance *locale*, c'est-à-dire celle séparant les deux points I_i et I_j qui caractérisent le point de contact.

Notons que le problème du repositionnement est en tous points semblable aux problèmes dynamiques du chapitre 5, puisqu'il s'agit de calculer les vitesses les plus proches possibles de la vitesse nulle, dans l'espace des vitesses, parmi celles qui satisfont les contraintes de repositionnement.

Dans notre implantation, les objets sont redimensionnées de façon que la largeur totale de la scène soit de 50 unités, et les paramètres de couplage sont $\epsilon_r = 0,0044$, $\epsilon_s = 0,01$, et $\epsilon_b = 0,023$. Ces valeurs permettent de ne pas voir les effets du repositionnement. Notons que, pour des scènes comportant des objets mobiles de tailles très diverses, il est possible, au cours de l'interaction, d'adapter ces valeurs à l'objet manipulé et observé, afin d'éviter que la taille de l'objet manipulé soit du même ordre de grandeur que la distance de sécurité. Dans les bases de données que nous avons testées, cependant, cela n'a pas été nécessaire.

Notons enfin que les repositionnements des objets ne correspondent pas à un phénomène physique, et peuvent donc ajouter de l'énergie dans le système. Afin d'éviter cela, il peut être utile par exemple de diminuer légèrement la vitesse des objets après un repositionnement (notons que ce problème ne se pose pas dans une simulation du premier ordre, ou quasi-statique, puisque les vitesses des objets sont nulles à tout instant).

De façon plus générale, le problème de repositionnement des objets s'apparente à un problème de stabilisation des contraintes, dont une solution classique est celle proposée par Baumgarte [Bau72]. Dans ce cas, cependant, les contraintes sont unilatérales. Par ailleurs, le problème plus important de stabilisation des contraintes unilatérales *de repositionnement* est grandement facilité par la détection de collisions continue.

6.3 Evaluation des algorithmes

Plusieurs bases de données ont été utilisées pour évaluer le simulateur dynamique fondé sur les algorithmes des chapitres précédents. Certaines d'entre elles étaient simplement destinées à vérifier le bon fonctionnement des fonctions de détection de collisions, alors que d'autres, correspondant à des cas industriels, visaient à évaluer le passage à l'échelle.

Pour ces évaluations, un PC doté d'un processeur pentium 1 GHz a été utilisé. La manipulation des objets était effectuée à l'aide d'une simple souris. Afin de faciliter l'interaction, l'utilisateur avait la possibilité de se déplacer dans la scène, et/ou d'afficher les objets non manipulés en transparence, au cours de la manipulation.

6.3.1 Premières évaluations

Dans un premier temps, nous avons simplement cherché à vérifier les algorithmes de détection de collisions et de dynamique, ainsi qu'à évaluer la qualité de l'interac-

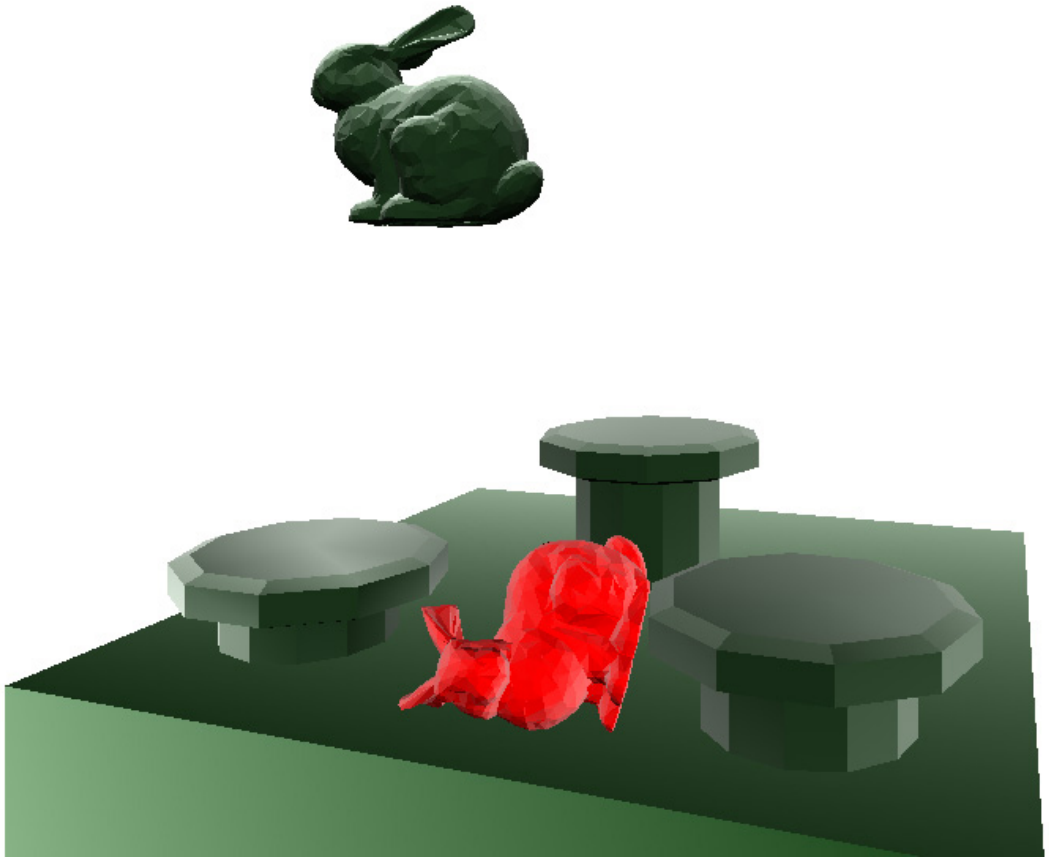


Fig. 6.2 – Première évaluation : l'utilisateur peut tester plusieurs situations classiques (positionnement d'objet, mouvements lents ou rapides entrainant des collisions, avec ou sans gravité).

tion offerte par une détection de collision continue. Pour cela, nous avons conçu deux applications simples.

Dans la première, visible dans la figure 6.2, un utilisateur peut manipuler un objet (le lapin sur le sol) dans un environnement statique (6000 faces en tout), et tester plusieurs types d'interaction classiques, comme le positionnement précis d'un objet, et les mouvements lents ou rapides entrainant des collisions.

La manipulation peut se faire dans un monde du *premier ou deuxième ordre*. Au premier ordre, quasi-statique, les objets n'ont pas d'accélérations, et ne changent de position que lorsque l'utilisateur exerce une action sur eux. Au deuxième ordre, qui correspond à notre réalité, les objets peuvent avoir des vitesses et des accélérations non nulles.

La seconde application, visible dans la figure 6.3, consiste à faire rebondir un modèle de chassis de voiture Scénic, fourni par Renault (29000 faces), afin de tester les algorithmes de calcul de mouvement contraint, ainsi que le modèle de friction formulé dans l'espace des mouvements (*cf* chapitre 5). Nous avons ainsi pu constater que, d'une

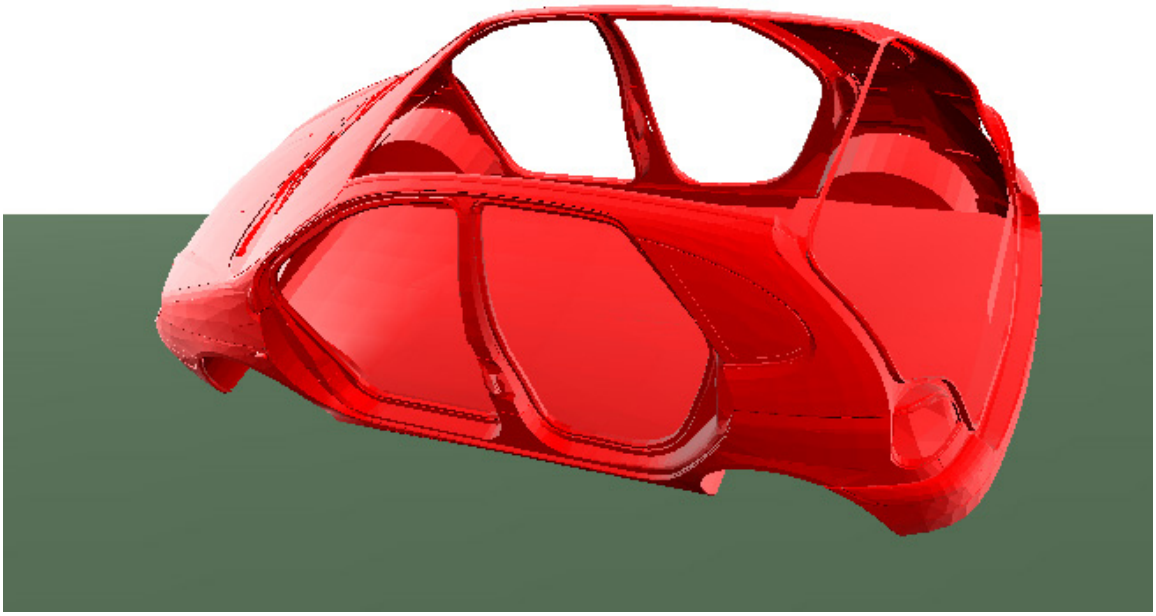


Fig. 6.3 – Un chassis de voiture, manipulé par l'utilisateur, est soumis à la gravité et à des frottements statiques et dynamiques.

part, utiliser un mouvement intermédiaire arbitraire pour la détection de collisions permet bien d'obtenir une simulation interactive réaliste (aux yeux de l'utilisateur) et que, d'autre part, notre modèle de friction semble convenir aux applications de simulation dynamique interactive.

6.3.2 Bases de données industrielles

Nous avons ensuite effectué des tests sur des bases de données provenant de cas industriels réels, fournis par Renault et Airbus-EADS. Ces tests ont permis de montrer la validité et l'efficacité des algorithmes de détection de collisions et de calcul du mouvement contraint dans un contexte industriel.

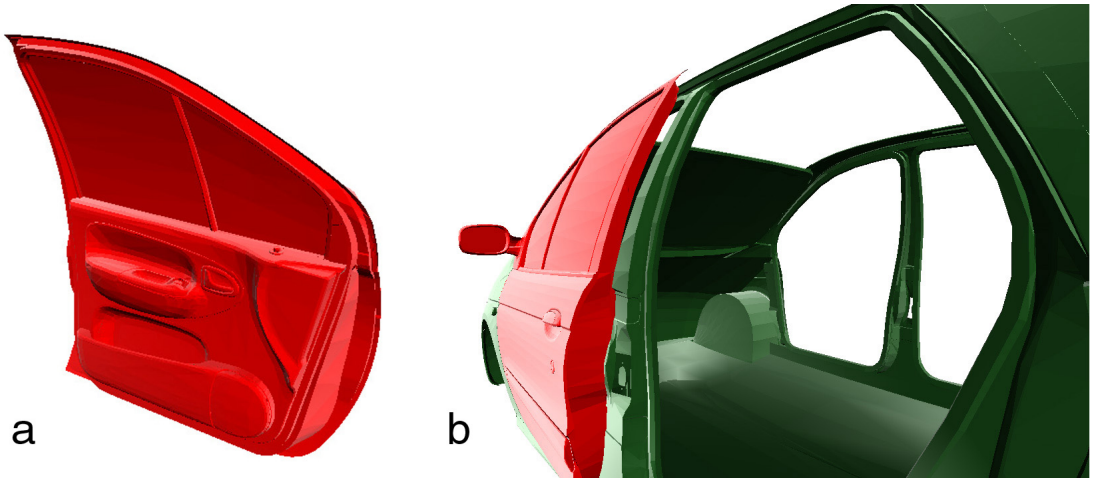


Fig. 6.4 – Positionnement interactif d'une portière. La détection de collisions continue permet à l'utilisateur de positionner simplement et très précisément la portière de la voiture, sans aucune interpénétration.

6.3.2.1 Positionnement d'une portière de voiture

Le premier test a consisté à positionner une portière de voiture Renault Scénic, visible dans la figure 6.4.a (16000 triangles), sur le chassis utilisé précédemment. Malgré un périphérique d'interaction peu adapté, il s'avère que la portière peut être placée très simplement et précisément. A chaque instant, le fait que la portière ne pénètre pas dans le chassis contribue grandement à augmenter le réalisme de la simulation et la précision de la manipulation. La figure 6.4.b montre la portière en train d'être positionnée par l'utilisateur.

6.3.2.2 Démontage d'un moteur de lève-vitre

Le deuxième test consistait à enlever un moteur de lève-vitre de l'intérieur d'une autre portière. La scène, comportant environ 20000 triangles, est visible dans la figure 6.5. Les tests ont montré que les algorithmes de détection de collisions continue et de calcul du mouvement contraint permettent à des utilisateurs non expérimentés avec le simulateur de sortir simplement le moteur de la portière à l'aide de la souris.

Plusieurs étapes du démontage sont visibles dans la figure 6.5. Les sphères jaunes matérialisent les différents points de contact.

6.3.2.3 Mat de réacteur d'avion

Un autre test consistait à démonter un élément d'une partie d'un modèle de mât de réacteur d'avion. La scène dans son ensemble, comportant environ 80000 triangles,

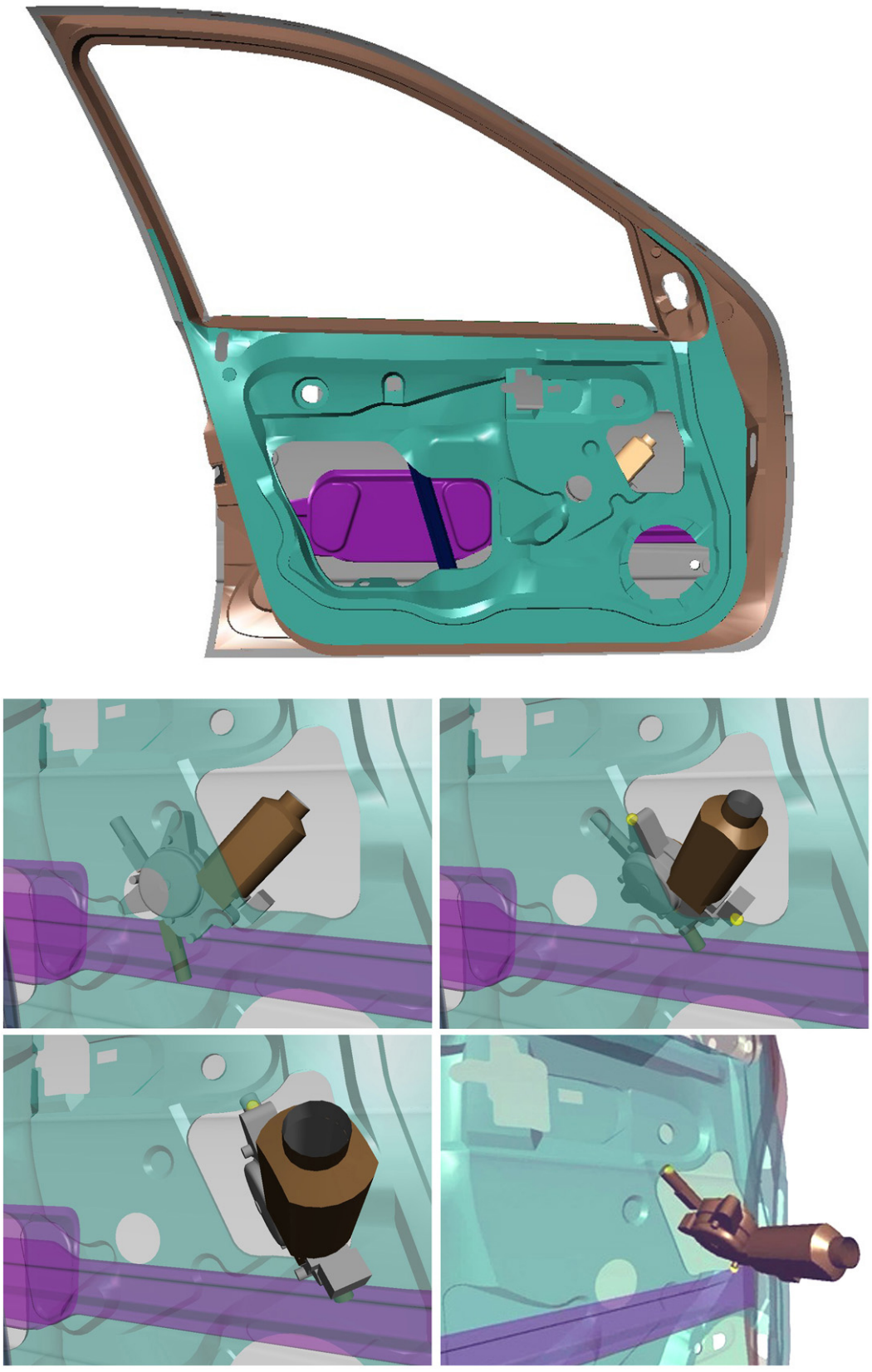


Fig. 6.5 – Le moteur du lève-vitre est facilement enlevé de la portière à l'aide d'une simple souris. Les sphères jaunes matérialisent les points de contact.

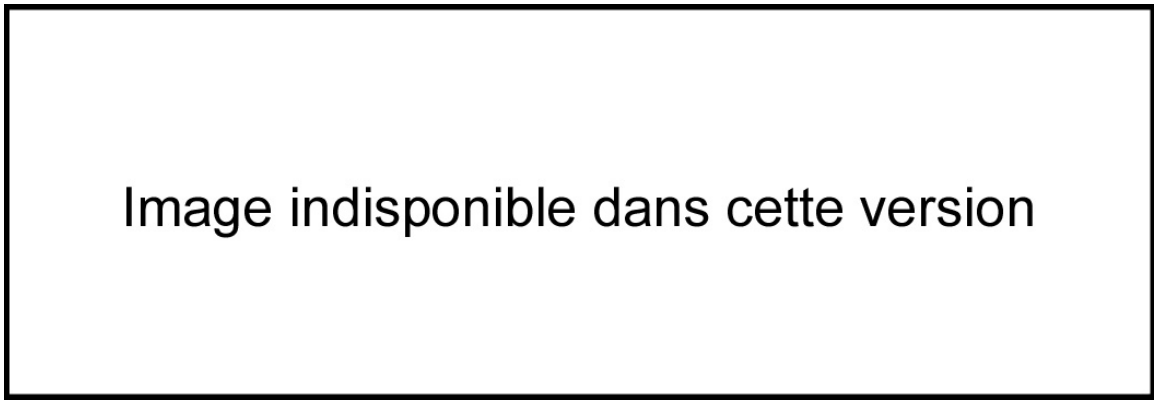


Fig. 6.6 – Une partie du mât de réacteur d'avion (modèles ©Airbus).

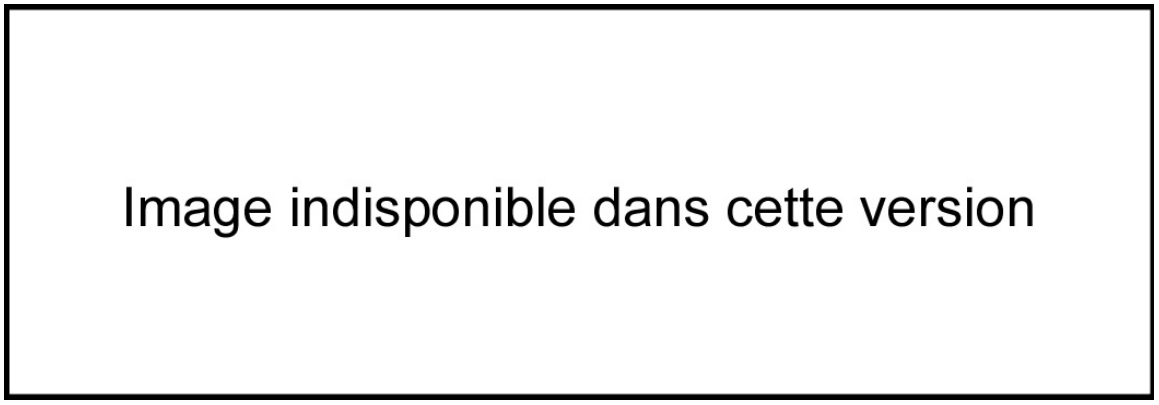


Fig. 6.7 – Mât de réacteur d'avion : agrandissement de la partie à démonter (modèles ©Airbus).

est visible dans la figure 6.6[2].

Un agrandissement de la partie à démonter est visible dans la figure 6.7. Pour distinguer l'objet manipulé, l'environnement est affiché en transparence.

En raison de la forme de l'environnement, il est nécessaire de faire pivoter l'objet pour l'extraire de l'environnement. Ceci est perceptible très vite dans le simulateur et permet de comprendre rapidement comment extraire l'objet. Le démontage est alors réalisé très simplement, en faisant glisser l'objet sur l'environnement, en une trentaine de secondes. Quatre étapes du démontage sont visibles dans le figure 6.8.

Encore une fois, il apparaît que les algorithmes de détection de collisions et de calcul de mouvement contraint permettent de réaliser facilement des tâches réalistes et complexes.

[2]Cette base de données étant confidentielle, nous ne sommes pour le moment pas autorisés à diffuser les figures 6.6, 6.7 et 6.8 dans la version électronique. Les images sont disponibles dans la version papier du mémoire. Veuillez nous en excuser.

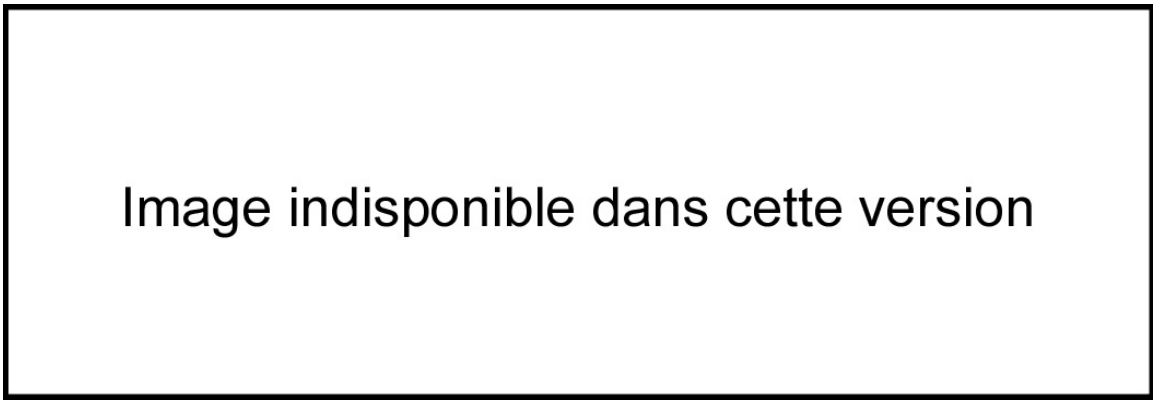


Fig. 6.8 – Quatre étapes du démontage (modèles ©Airbus).

6.3.3 Intégration sur le plan de travail virtuel

Nous avons pu effectuer plusieurs tests sur le plan de travail virtuel de l'INRIA, visible[3] dans la figure 6.9. Brièvement, le plan de travail virtuel est un système de visualisation et d'interaction avec des environnements virtuels permettant une très grande immersion, grâce au grand angle de vision et aux différents périphériques d'interaction. Un système de stéreoscopie active, couplé à un capteur electromagnétique fixé sur les lunettes à cristaux liquides, permet de coordonner les environnements réel et virtuel. Il est ainsi possible, en déplaçant la tête, d'observer les objets virtuels sous plusieurs points de vue, comme on le ferait en réalité.

Le périphérique d'interaction que nous avons utilisé est un stylo équipé d'un capteur electromagnétique. Le capteur, en transmettant la position et l'orientation du stylo (six degrés de liberté au total) permet à l'utilisateur d'interagir avec les objets virtuels. Un rayon virtuel, affiché en permanence dans le prolongement du stylo réel, permet de sélectionner les objets à déplacer.

Deux métaphores d'interaction ont été testées. Dans la première, *incrémentale* [ZR01], tout déplacement du stylo est immédiatement considéré comme un *mouvement intentionnel* de l'objet. Ce mouvement intentionnel est envoyé au simulateur qui calcule le mouvement contraint de l'objet et détecte les collisions. Cette métaphore permet de manipuler simplement l'objet lorsqu'il n'est pas en contact avec l'environnement, mais introduit un décalage entre le rayon virtuel et l'objet dès lors que celui-ci entre en contact avec l'environnement. Ce décalage peut être ensuite conservé, même lorsque l'objet n'est plus en contact avec l'environnement.

La figure 6.10 montre ce phénomène. A l'instant initial, le stylo est en position S_0 et l'objet manipulé est en position C_0. Ensuite, le stylo est en position S_1, mais le mouvement intentionnel $\mathcal{M}_i$ résultant du mouvement du stylo, est transformé en le mouvement contraint $\mathcal{M}_c$, introduisant ainsi un décalage entre le rayon virtuel et

[3]Les photographies représentant le plan de travail virtuel sont des montages : les objets virtuels ont été réincrustés dans l'image afin de mieux comprendre ce que voit l'utilisateur.

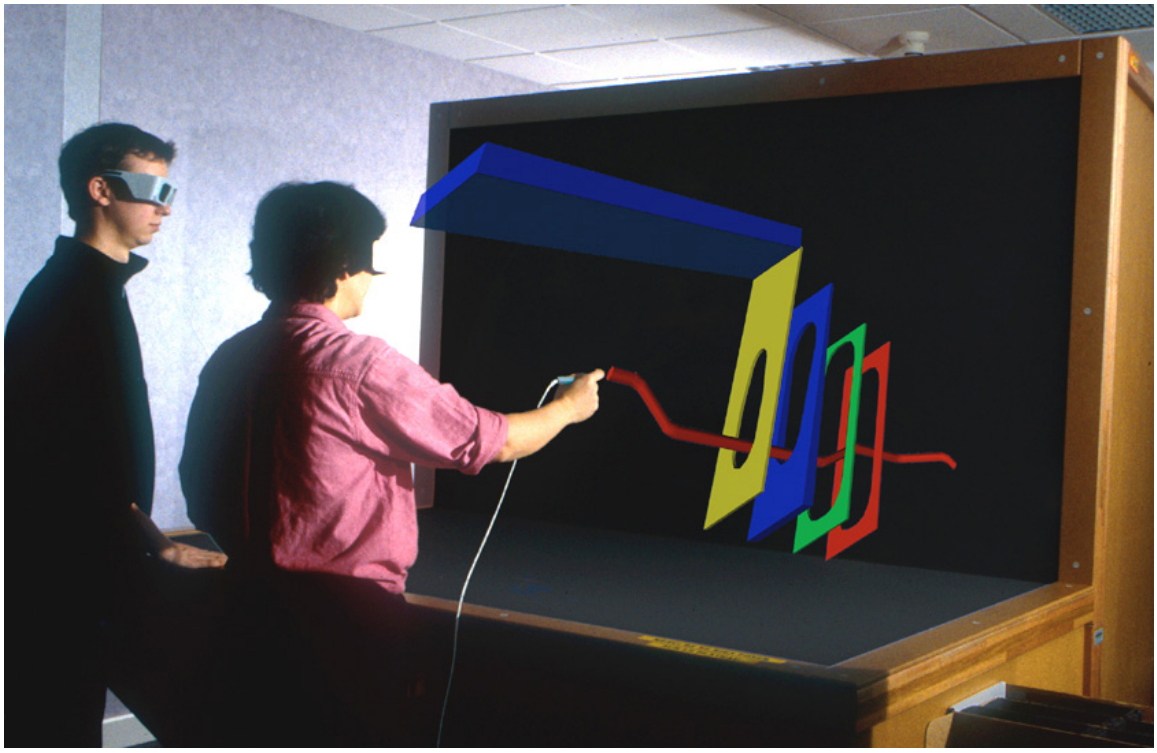


Fig. 6.9 – Intégration de CONTACT Toolkit sur le plan de travail virtuel.

l'objet manipulé en C_1. Si, enfin, l'utilisateur effectue un mouvement de translation vers le haut (position du stylo S_2), le cube rompt le contact avec le sol mais le décalage est conservé (position du cube C_2).

La deuxième métaphore d'interaction, dite du *ressort virtuel* [ZR01, FTBAB00, Fra01], remplace le rayon virtuel par un ressort agissant sur les six degrés de liberté de l'objet, et le contraignant à chaque instant à revenir à la position qu'il avait, lors de sa sélection, relativement au stylo. Cette métaphore, que l'on pourrait qualifier de *pseudo-haptique*, puisqu'elle substitue des informations visuelles (la déformation du ressort) aux informations haptiques, permet de supprimer le décalage entre le rayon virtuel et l'objet, lorsque celui-ci n'est plus en contact avec l'environnement virtuel.

Si l'on reprend la même succession de mouvements que dans la figure 6.10 (*i.e.* en conservant les positions S_0, S_1 et S_2), le cube à la fin du deuxième mouvement est à l'extrêmité du ressort, en C_2, qui a repris sa position de repos. Ce phénomène est représenté dans la figure 6.11.

6.4 Simulation interactive avec retour d'efforts

Nous avons également testé les algorithmes exposés dans ce mémoire avec plusieurs périphériques de retour d'efforts. Pour cela, quatre configurations ont été

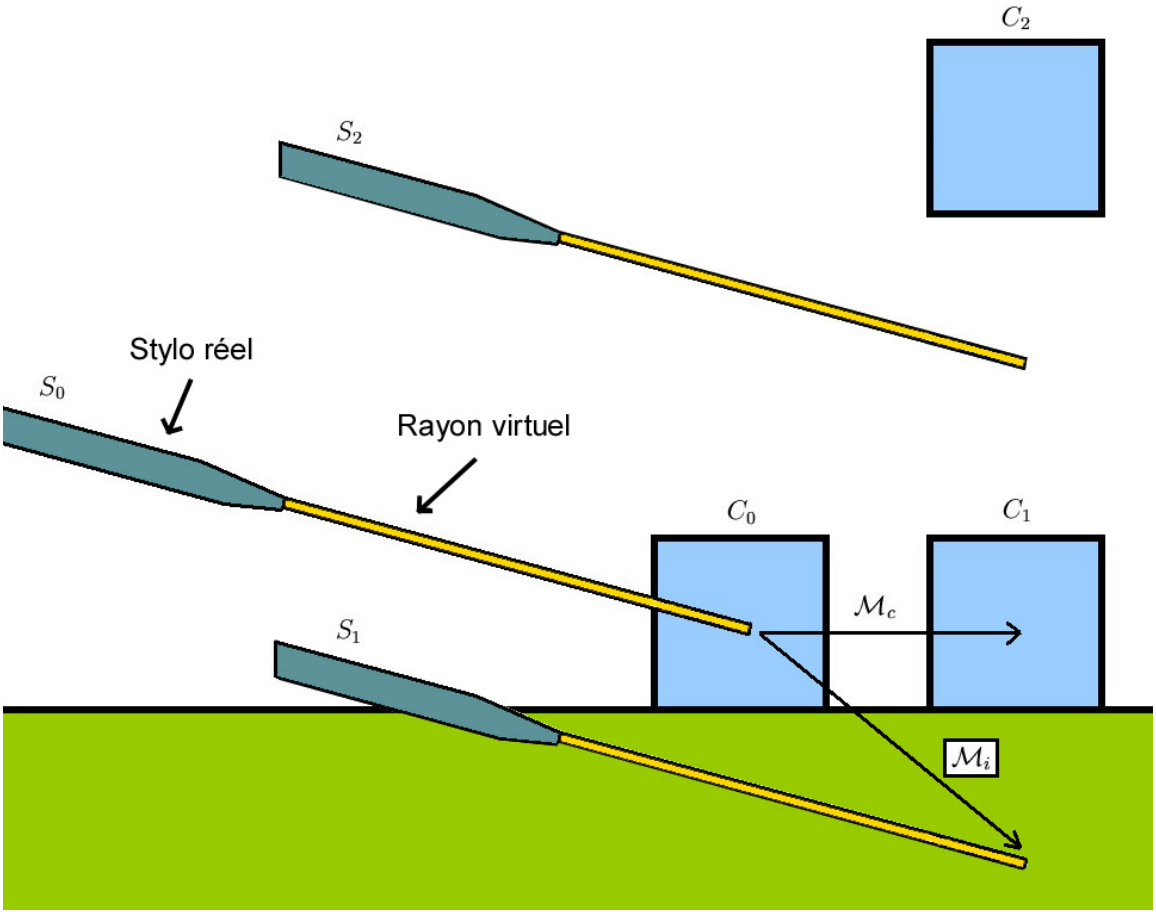


Fig. 6.10 – Sur le plan de travail virtuel, une métaphore d'interaction incrémentale entraîne un décalage de plus en plus grand entre le périphérique d'interaction et l'objet manipulé lorsque celui-ci est en contact avec l'environnement.

utilisées, le bras *Phantom* de *Sensable Technologies* [MS94], la *poignée haptique* [Lec01, LMBLCCG02] et le bras *Virtuose* du CEA, et le SPIDAR, développé par Sato [BIS00] et adapté sur le plan de travail virtuel de l'INRIA.

6.4.1 Premières expériences

6.4.1.1 Couplage du simulateur à un bras de retour d'efforts

Le bras Phantom a été utilisé par Anatole Lécuyer afin de développer un prototype de système de montage/démontage pour Airbus-EADS. Seuls les algorithmes de détection de collisions continue ont été utilisés pour le simulateur dynamique [LKCGC01]. Le calcul du mouvement contraint des objets était effectué par les algorithmes d'Anatole. En particulier, l'absence de retour d'efforts en rotation l'ont conduit à développer la métaphore de la sphère virtuelle, pour manipuler les objets en rotation et ressentir des couples, en les convertissant en forces [LKCGC01].

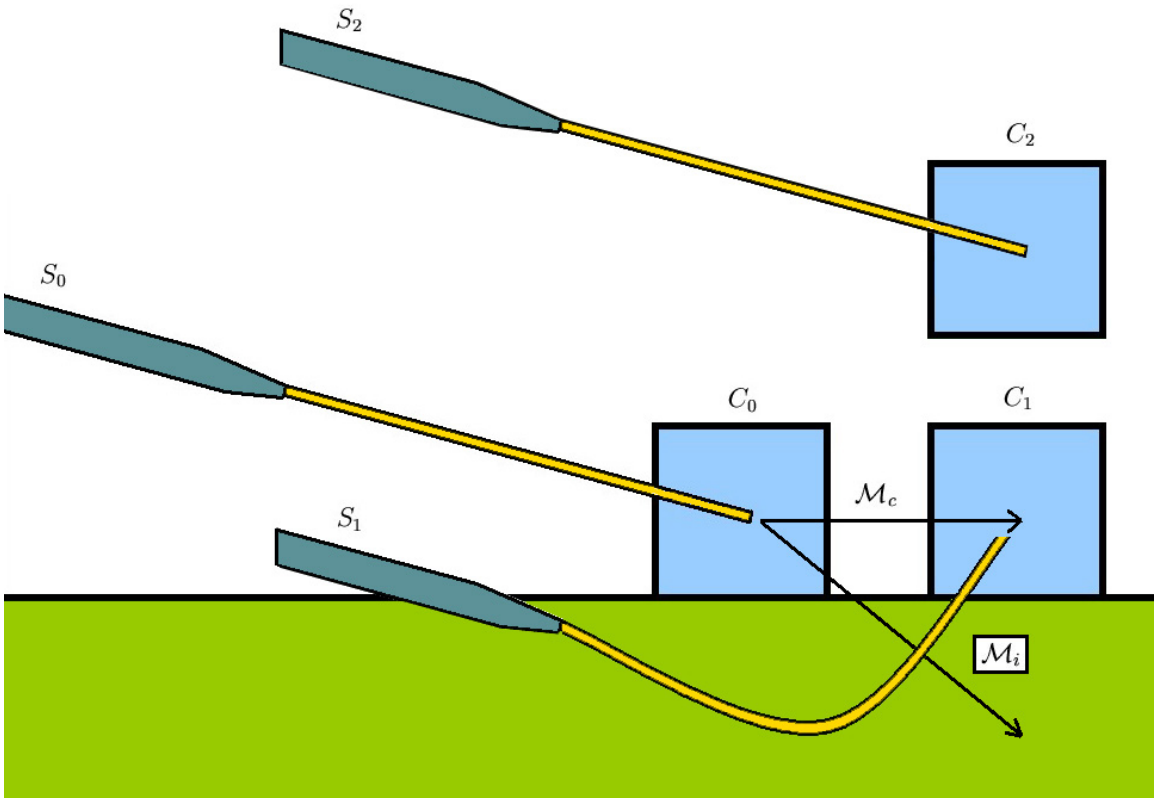


Fig. 6.11 – La métaphore du ressort permet de résoudre le problème du décalage observé avec la métaphore incrémentale.

Des tests informels réalisés auprès d'opérateurs d'Airbus ont permis de montrer l'intérêt pratique du démonstrateur. La figure 6.12 montre un utilisateur procédant à l'insertion d'un tuyau dans un modèle simplifié de tuyère.

6.4.1.2 Couplage du simulateur à une poignée haptique

Les algorithmes de détection de collisions continue ont également été utilisés pour réaliser une expérience visant à déterminer l'apport d'informations visuelles, auditives et haptiques dans une tâche d'insertion sur le plan de travail virtuel [LMBLCCG02]. Il ressort de cette étude qu'aucune des informations supplémentaires (*i.e.* auditive ou haptique) n'a un impact positif sur le temps de réalisation de la tâche, mais que le rendu haptique est apprécié par les sujets, qui le trouvent utile, agréable et apte à améliorer le réalisme de la simulation.

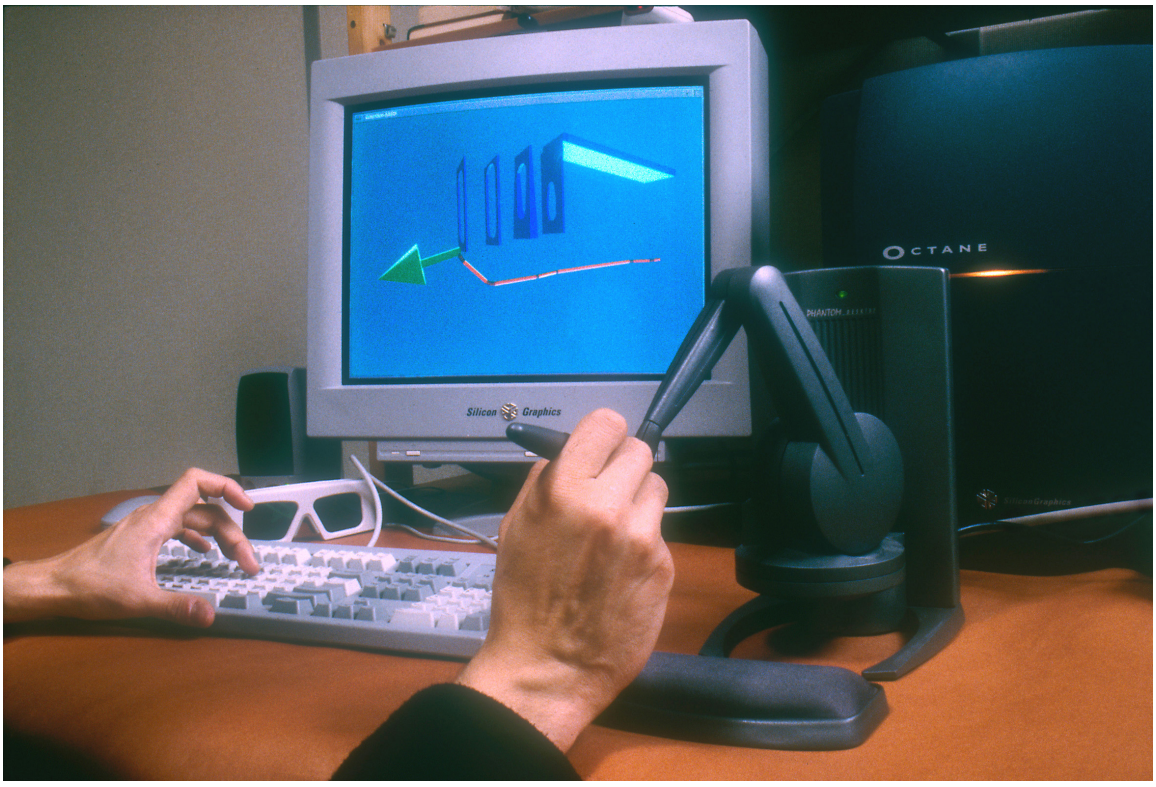


Fig. 6.12 – Couplage des algorithmes de détection de collision à un bras Phantom^{TM}.

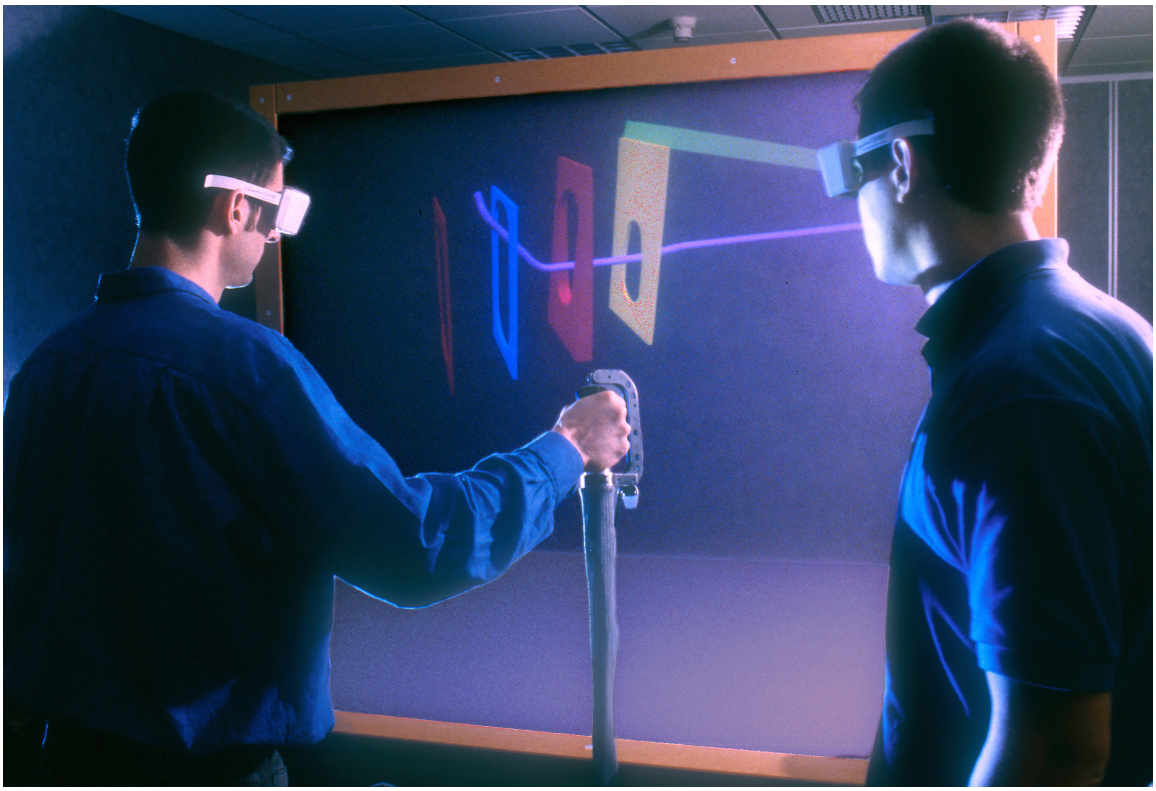


Fig. 6.13 – Couplage des algorithmes de détection de collision à une poignée haptique.

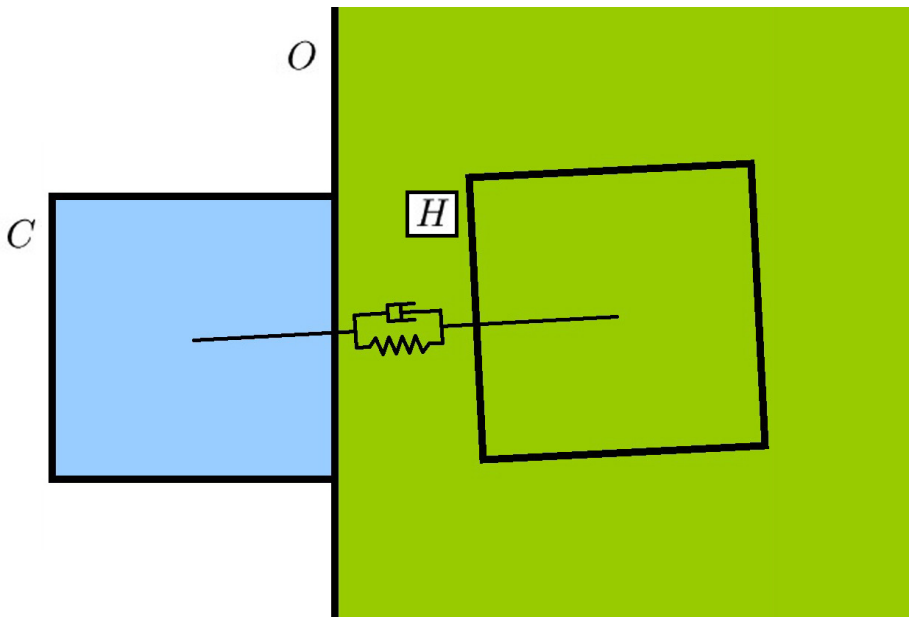


Fig. 6.14 – Le couplage virtuel est établi sous la forme d'un ressort amorti reliant l'objet virtuel C et la représentation du périphérique haptique H dans le monde virtuel.

6.4.2 Intégration du retour d'efforts dans CONTACT Toolkit

Dans ces premières expériences, seuls les algorithmes de détection de collisions ont été utilisés, et des modèles de calcul de mouvement contraint et de retour d'efforts *ad hoc* ont été employés.

Cependant, nous avons également testé l'ensemble du simulateur CONTACT Toolkit, comprenant les algorithmes de calcul de mouvement contraint, avec des périphériques haptiques. Pour cela, nous avons utilisé une méthode de couplage classique pour relier le périphérique haptique aux algorithmes de simulation dynamique : la méthode du couplage virtuel, proposée par Colgate *et al.* [CSB95].

Supposons dans un premier temps que les mondes virtuel et réel soient superposables, et de même échelle. C'est le cas par exemple sur le plan de travail virtuel. En quelques mots, la méthode du couplage virtuel consiste à attacher un ressort virtuel amorti entre l'objet virtuel et le périphérique haptique. Ainsi, à chaque instant, l'objet virtuel et le périphérique haptique tentent de parvenir à la même position. Plus précisément, à chaque instant, le mouvement intentionnel de l'objet manipulé est de rejoindre le périphérique haptique (commande en vitesse ou en accélération selon l'ordre du monde). Inversement, la force exercée par le périphérique haptique sur la main de l'utilisateur tend à faire revenir celle-ci à la position de l'objet.

Plus généralement, lorsque les mondes virtuels et réels ne coïncident pas (par exemple lorsque un bras à retour d'efforts est utilisé sur une station de travail sans stéréovision), le couplage virtuel s'effectue entre l'objet virtuel et *la représentation du périphérique haptique dans le monde virtuel* (le *god-object*). Le périphérique haptique

et sa représentation virtuelle ne coïncident que lorsque les mondes virtuel et réel coïncident eux aussi.

Le principe du couplage virtuel est visible dans la figure 6.14. Dans cet exemple, le mouvement du cube virtuel C est contraint sur l'obstacle O. En raison de la raideur finie du ressort, la représentation du périphérique haptique dans le monde virtuel H a pénétré l'environnement. Le couplage virtuel, en exerçant une action sur le périphérique haptique, simule la force de réaction, créée par le contact entre l'objet et l'obstacle, que ressentirait l'utilisateur en manipulant un objet réel dans une configuration similaire.

Le principe du couplage virtuel est extrêmement simple à mettre en oeuvre. Notons x_v la coordonnée de l'objet virtuel suivant l'un de ses degrés de liberté, et notons x_h la coordonnée correspondante pour la représentation du périphérique haptique dans le monde virtuel. Alors la force exercée sur l'objet virtuel, suivant ce degré de liberté, est

$$f_v = k_v \left(x_h - x_v\right) + b_v \left(\dot{x}_h - \dot{x}_v\right) \tag{6.3}$$

où k_v et b_v sont les constantes de couplage pour l'objet virtuel, et où $\dot{x}_h$ et $\dot{x}_v$ sont les dérivées respectives de x_h et x_v par rapport au temps.

De même, la force exercée sur le périphérique haptique suivant ce degré de liberté est

$$f_h = k_h \left(x_v - x_h\right) + b_h \left(\dot{x}_v - \dot{x}_h\right) \tag{6.4}$$

où k_h et b_h sont les constantes de couplage pour le périphérique haptique.

Notons qu'il est possible de modifier cette dernière expression afin d'améliorer la stabilité du périphérique haptique en réduisant en permanence sa vitesse. La force exercée sur le périphérique devient ainsi :

$$f_h = k_h \left(x_v - x_h\right) - b_h \dot{x}_h \tag{6.5}$$

Ceci rajoute cependant artificiellement du frottement dans le rendu haptique, même lorsqu'il n'y en a pas dans la simulation.

L'utilisation de CONTACT Toolkit pour la simulation interactive avec retour d'efforts a été testée à l'aide de deux configurations : le bras Virtuose du CEA et le SPIDAR, adapté au plan de travail virtuel. Une version du Virtuose, possédant six degrés de liberté en entrée et en sortie, est visible dans la figure 6.16. Nous avons procédé à des tests simples seulement sur le Virtuose, qui nous ont permis de constater le bon fonctionnement du couplage virtuel. L'essentiel des tests de CONTACT Toolkit pour le rendu haptique a été effectué sur le SPIDAR.

Le SPIDAR est un périphérique haptique à base de cables et de moteurs pas à pas, facilement reconfigurable, qui a été adapté sur le plan de travail virtuel de l'INRIA. La figure 6.15 montre la position des cables sur le plan de travail virtuel. Les flèches pointent sur deux des quatre moteurs. L'utilisateur place son index au point d'intersection des cables, comme représenté dans la figure 6.17, et les moteurs,

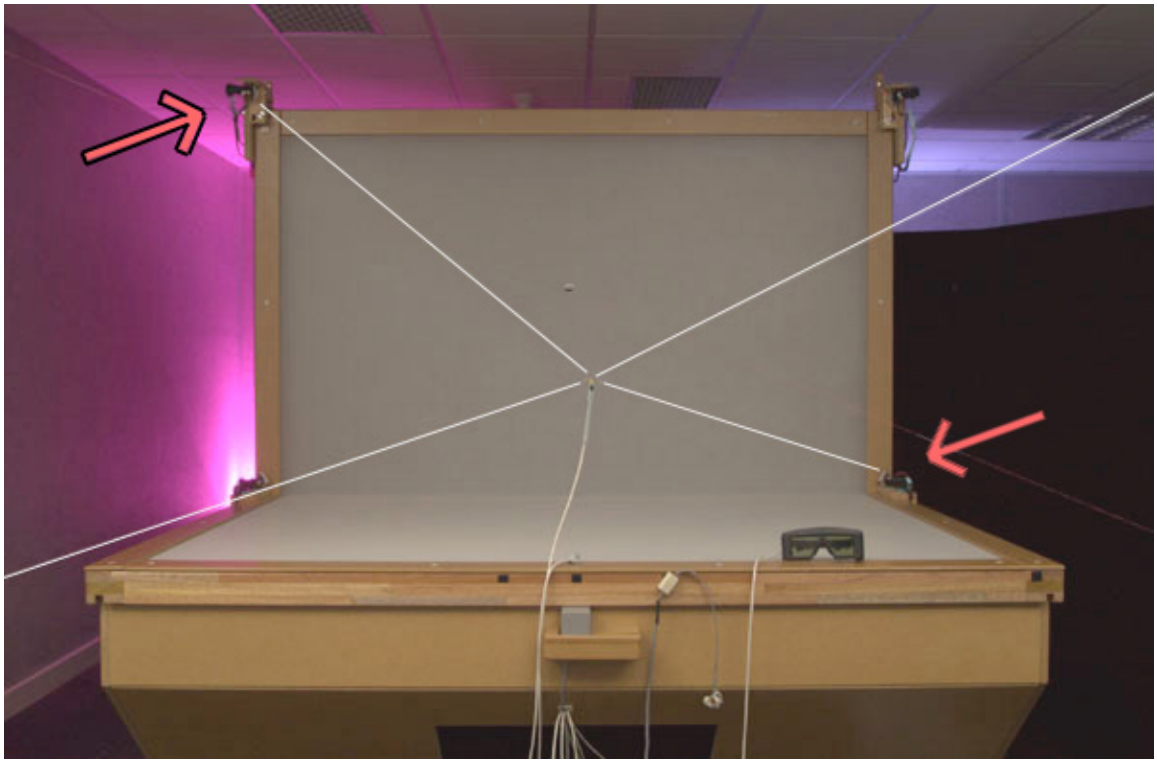


Fig. 6.15 – Position des cables du SPIDAR sur le plan de travail virtuel.

en enroulant ou déroulant les cables, exercent une force sur le doigt de l'utilisateur. La figure 6.18 représente un utilisateur du SPIDAR dans une situation typique.

La version du SPIDAR représentée sur les figures 6.15, 6.17 et 6.18 dispose de trois degrés de liberté en entrée (lecture de la *position* du doigt seulement, et non de son orientation), ainsi que de trois degrés de liberté en sortie (l'utilisateur peut ressentir une force, mais pas de couple). Dans cette version du SPIDAR, en effet, il n'est possible de contrôler qu'un seul point, le point d'intersection des cables, et l'objet ne peut être commandé qu'en translation, les rotations étant obtenues en le faisant basculer sur les obstacles.

Cette première version nous a toutefois permis de réaliser quelques expériences démontrant le confort visuel obtenu grâce à la détection de collisions continue. L'une des toutes premières expériences réalisées consistait à manipuler un cube mobile et à l'amener en contact avec un autre cube. La simulation, effectuée dans un monde du premier ordre (sans accélérations), s'est avérée extrêmement réaliste et robuste, en particulier pour les transitions entre types de contact, par exemple lorsque le cube manipulé passait d'une face à l'autre du cube statique (*cf* figure 6.19). Dans cet exemple, en effet, le point A ne pénètre jamais dans le coin B du cube statique, alors qu'il le ferait pour une méthode de détection de collisions discrète. Avec une méthode discrète, le simulateur ferait vraisemblablement sortir le point A par la droite du cube statique après l'avoir laissé pénétrer par le haut. Comme prévu, la détection de collisions continue permet d'éviter de tels comportements incohérents, déjà évoqués au chapitre 1

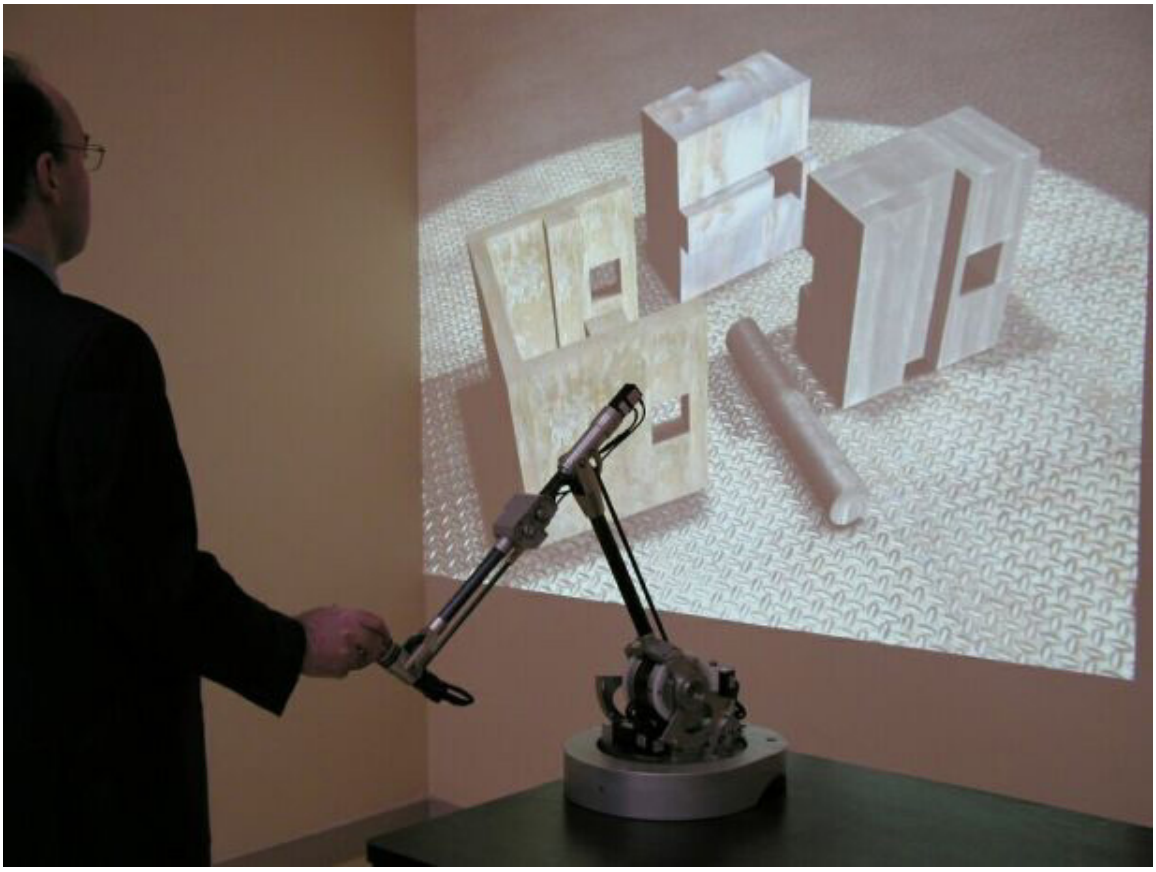


Fig. 6.16 – Le bras à retour d'efforts Virtuose^{TM}.

(figure 1.11).

La deuxième expérience notable réalisée sur cette version du SPIDAR a consisté à *toucher* des objets virtuels complexes. Pour cela, l'objet manipulé était un très petit cube (quelques millimètres), qui suivait les déplacements en translation de l'index de l'utilisateur. L'utilisateur pouvait alors, en amenant le petit cube en contact avec l'objet statique, avoir l'impression de toucher les objets. Un des objets statiques testés dans cette expérience est visible dans la figure 6.20. Il s'agit d'un modèle de volant fourni par Renault, qui comporte environ 11000 triangles. Un aspect particulièrement appréciable de la précision autorisée par la détection de collisions continue était sans doute d'avoir l'impression de toucher précisément les détails du volant, en particulier le logo du constructeur et la rainure présentes à l'avant du volant.

Bien qu'il soit possible de manipuler un objet en n'agissant directement que sur les trois degrés de liberté en translation, en s'aidant des obstacles pour le faire pivoter, il a été choisi de combiner le stylo précédemment utilisé et le SPIDAR. Les cables ont ainsi été fixés sur le stylo, manipulable librement par l'utilisateur, pour offrir six degrés de liberté en entrée. La fixation des cables au stylo a été effectuée à l'aide d'un élastique afin de permettre un démontage aisé. La fixation est visible dans la figure 6.21.

Bien sûr, le retour d'efforts ne s'effectue toujours que sur les trois degrés de liberté

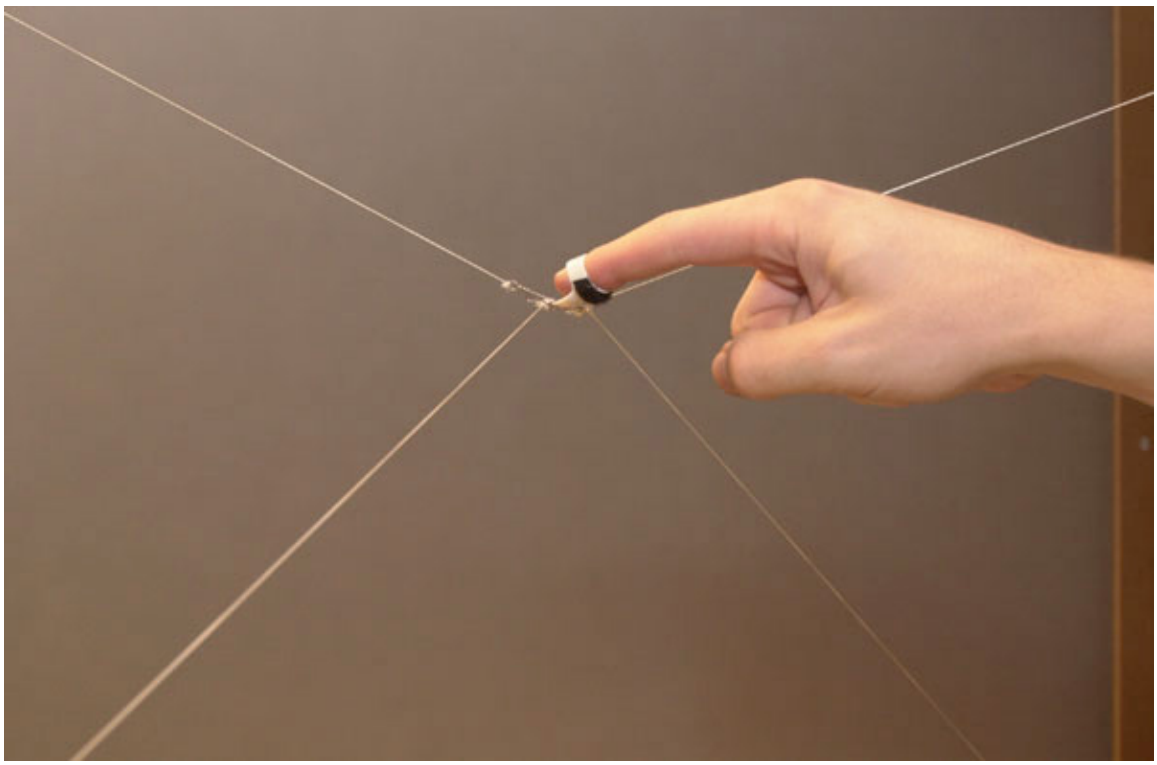


Fig. 6.17 – SPIDAR : Attachement des cables à l'index de l'utilisateur.

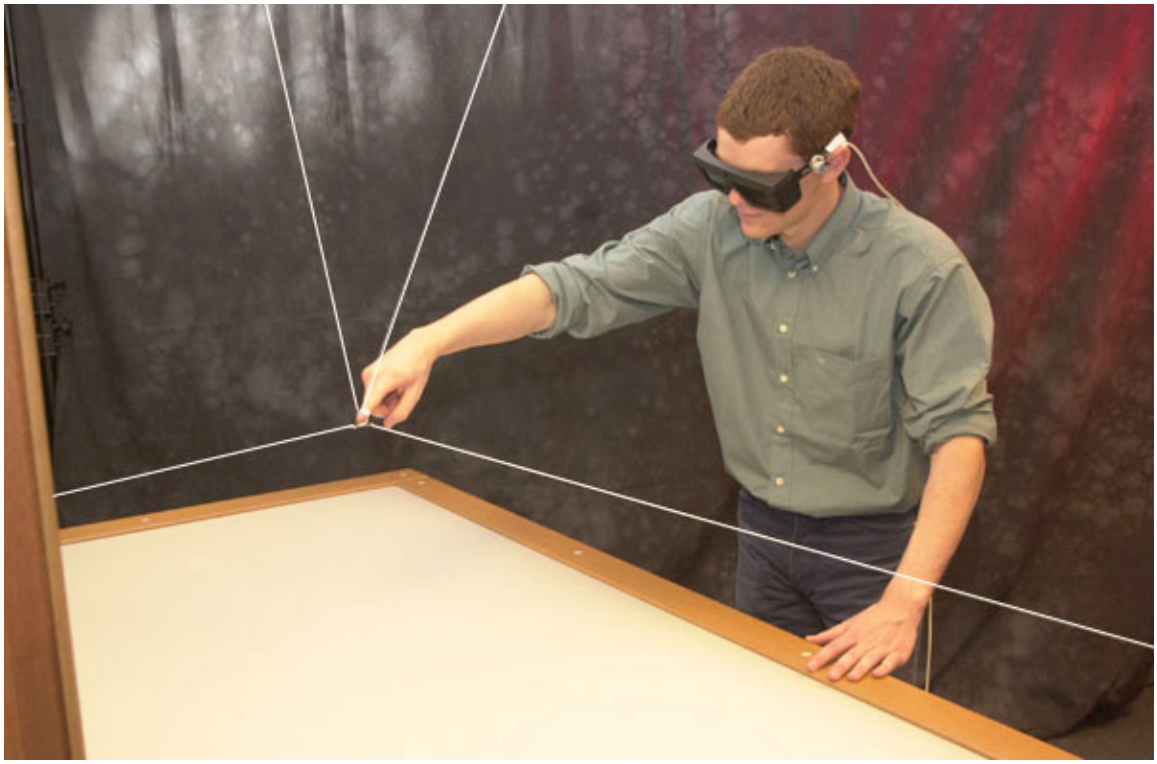


Fig. 6.18 – Utilisation du SPIDAR sur le plan de travail virtuel.

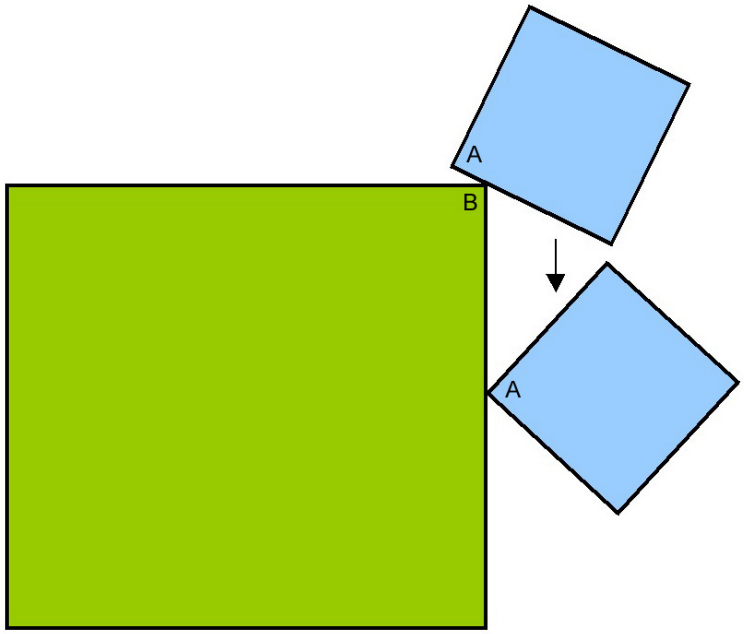


Fig. 6.19 – Les transitions entre les différents types de contact se réalisent de façon robuste.

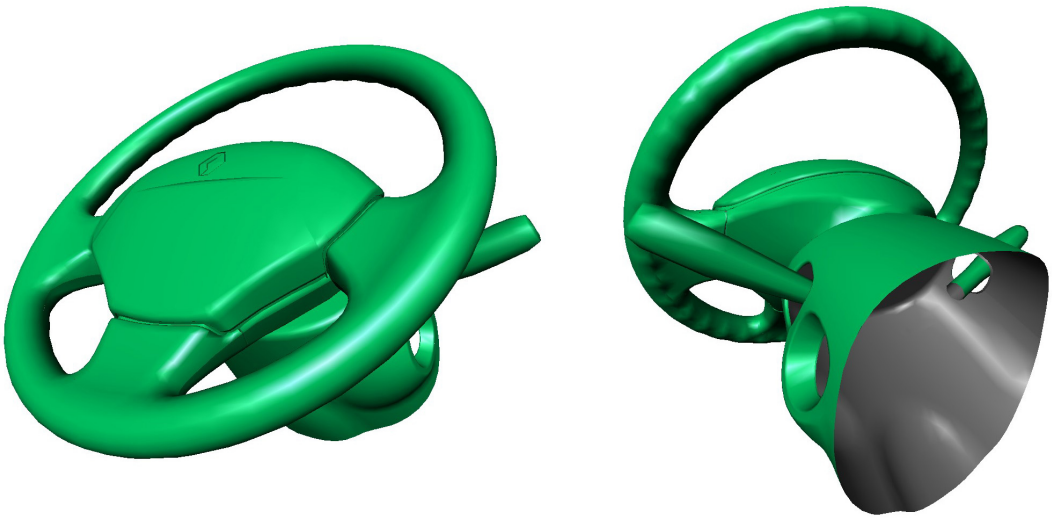


Fig. 6.20 – Modèle de volant utilisé pour les premiers tests de CONTACT Toolkit sur le SPIDAR.

en translation. Grâce à ce système, il est possible de réaliser des tâches complexes, comme l'insertion du moteur du lève-vitre dans la portière. Ceci est visible dans la figure 6.22.

Une autre version du SPIDAR, en cours d'adaptation sur le plan de travail virtuel,

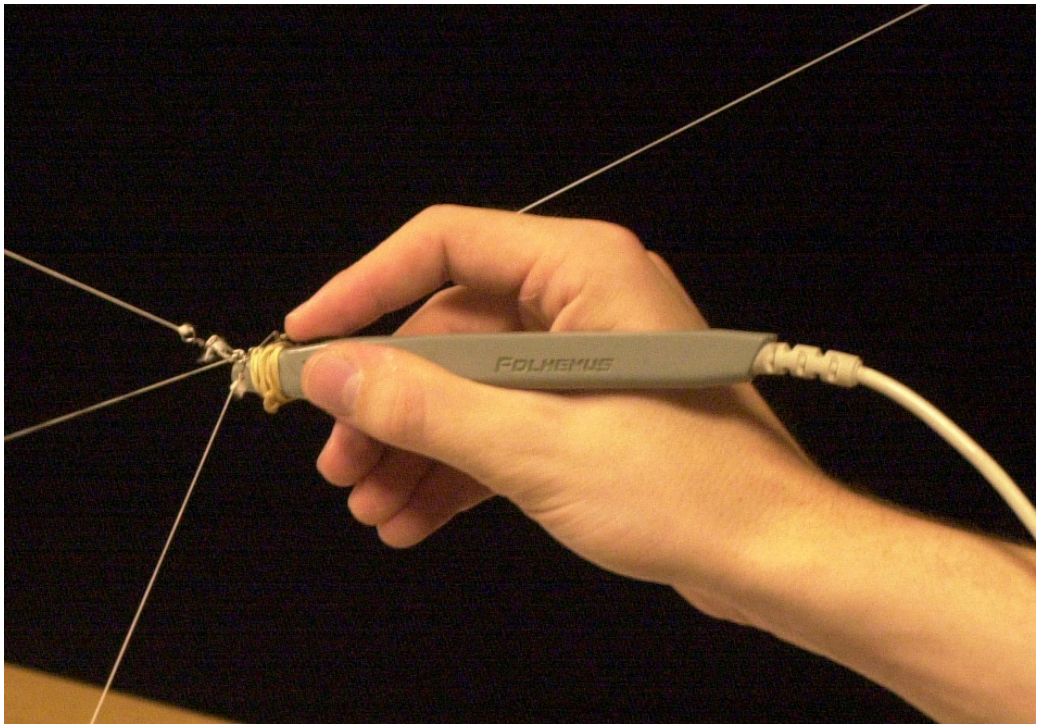


Fig. 6.21 – Fixation du stylo au SPIDAR.

double les cables et les moteurs pour permettre d'interagir avec deux points en même temps, offrant ainsi cinq degrés de liberté en entrée et sortie lors de la manipulation d'un objet rigide. En effet, les six degrés de liberté (trois pour chaque point) ne sont pas accessibles lorsqu'un objet rigide est manipulé : il faut retirer un degré de liberté à cause de la contrainte d'appartenance des deux points haptiques au même objet rigide. Plus précisément, il n'est pas possible de faire pivoter l'objet autour de l'axe passant par les deux points.

6.5 Discussion et conclusion

Dans ce chapitre, nous avons présenté la façon de coupler les algorithmes de détection de collisions à ceux permettant de calculer le mouvement contraint des objets. Ces algorithmes ont été implantés et combinés pour former CONTACT Toolkit, une librairie C++ portable et utilisable en particulier sur les environnements Windows, Unix et Linux.

Nous avons également présenté plusieurs applications du simulateur dynamique, dans des contextes variés et suffisamment complexes pour valider les algorithmes introduits dans ce mémoire. Certaines applications ont notamment permis de montrer la viabilité de nos algorithmes dans un contexte industriel. C'est le cas par exemple des bases de données fournies par Renault et Airbus-EADS.

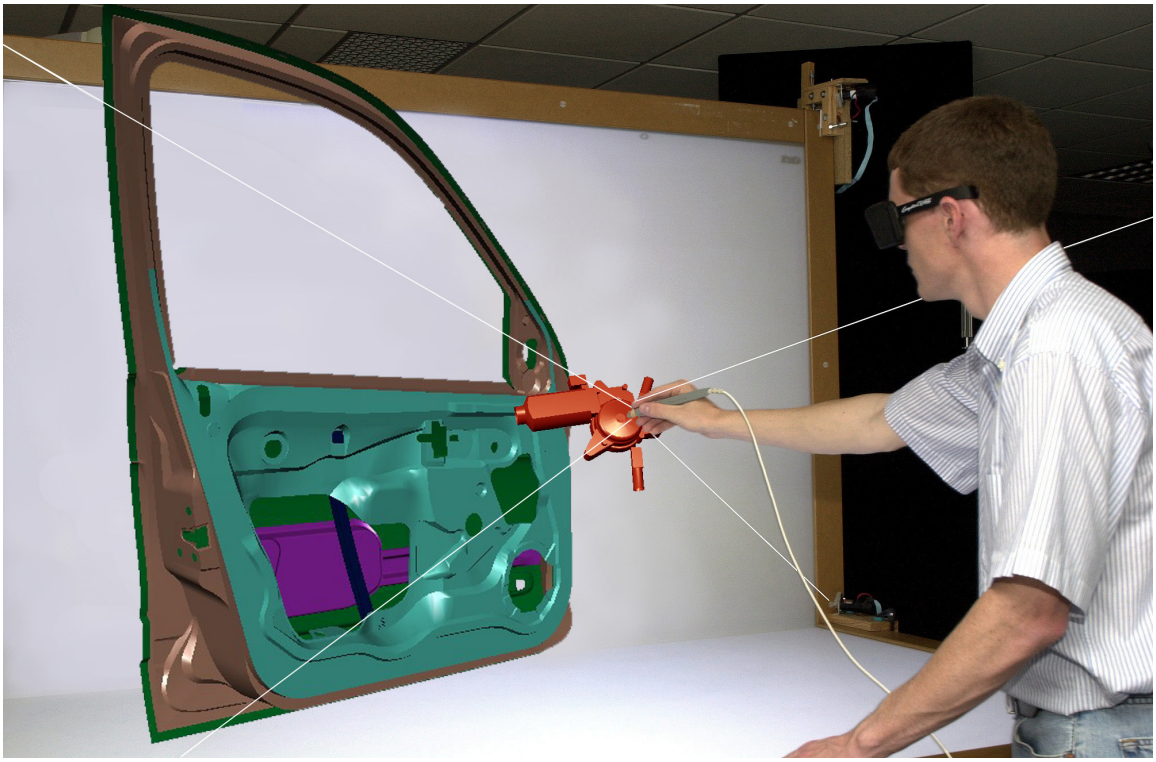


Fig. 6.22 – Utilisation de CONTACT Toolkit sur le SPIDAR.

De façon intéressante, la simulation du retour d'efforts *ne semble pas nécessaire* pour réaliser des tâches complexes (montage/démontage), notamment grâce à la possibilité qu'ont les objets de glisser sur les plans de contact détectés par les algorithmes continus. Ainsi, une station de travail munie d'une simple souris permet de résoudre certains problèmes industriels. Une évaluation plus approfondie du simulateur est en cours chez Renault et Airbus-EADS.

Remarquons, aussi, que pour toutes les tâches qui ne nécessitent pas la simulation des accélérations des objets (certaines tâches de montage/démontage, de planification de mouvement, de vérification de l'existence d'un chemin, de modélisation...), il semble tout à fait préférable d'effectuer la manipulation dans un monde du premier ordre, *i.e.* sans inertie.

Bien que le retour d'efforts ne semble pas nécessaire, il est tout à fait souhaitable dans certaines situations, afin par exemple d'améliorer le réalisme de la simulation et d'accroître l'immersion de l'utilisateur. Nous avons montré que les algorithmes introduits dans ce mémoire peuvent être utilisés très simplement pour la simulation interactive avec retour d'efforts.

En particulier, les algorithmes de détection de collisions continue ont été utilisés par Anatole Lécuyer pour établir un premier démonstrateur haptique de montage/démontage, ainsi que pour l'évaluation de la poignée haptique du CEA, et le simulateur complet, incluant les algorithmes de calcul de mouvement contraint, a été

testé avec succès sur le bras à retour d'efforts du CEA, le Virtuose, ainsi que sur la combinaison du SPIDAR et du plan de travail virtuel.

Nous pensons que la détection de collisions continue contribue grandement à la qualité du retour d'efforts pour deux raisons :

- la première est la *stabilité accrue* de la boucle haptique. Nous avons en effet vu au chapitre 1 qu'une cause importante d'instabilité peut être due aux interpénétrations intervenant entre les objets virtuels (figure 1.12). Ainsi, dans le cas du classique *peg-in-a-hole*, par exemple, une pénétration du bâton d'un côté de l'orifice à un instant t (position P_t) entraîne une force importante pour supprimer l'interpénétration. Cette force conduit souvent à une pénétration plus importante du côté opposé de l'orifice à l'instant suivant $t + 1$ (position P_{t+1}), qui crée elle-même une force de réaction plus importante que la précédente. Cette oscillation, en s'amplifiant, entraîne l'instabilité de la simulation [GMELM00]. Grâce à la détection de collisions continue, au contraire, le bâton virtuel reste dans l'orifice et les éventuelles instabilités de la simulation proviennent seulement de l'utilisation du ressort employé dans la méthode du couplage virtuel.
- La seconde raison pour laquelle la détection de collisions continue contribue à la qualité du retour d'efforts est *physiologique*. Il apparaît en effet que la plupart des utilisateurs attachent une grande importance à *l'image*, et privilégient ainsi le retour visuel sur le retour d'efforts lors de leur estimation de la raideur haptique du simulateur [SBB96]. Le fait que les objets virtuels ne s'interpénètrent pas et interagissent de façon très réaliste et intuitive accroit leur rigidité visuelle et augmente ainsi la raideur perçue du simulateur. Ceci a été particulièrement visible dans l'expérience du volant.

Ces deux facteurs conduisent ainsi à une mise en oeuvre simple et efficace du retour d'efforts.

Pour conclure ce mémoire, nous allons maintenant rappeler brièvement les résultats introduits dans les chapitres précédents, et tenter de mettre en avant plusieurs points qui nous semblent d'importance pour la suite des travaux.

Chapitre 7

Conclusion

Les travaux présentés dans ce mémoire ont été essentiellement motivés par la constatation des problèmes inhérents aux méthodes de détection de collisions discrètes. Nous avons vu en effet que des telles méthodes pouvaient notamment manquer de réalisme, de cohérence, et conduire à des instabilités dans la simulation.

Aussi, dans les chapitres précédents, nous avons proposé la notion de *mouvement intermédiaire arbitraire* pour interpoler les positions successives des objets rigides et détecter des collisions en continu efficacement. Essentiellement, nous avons dans un premier temps cherché à réduire le *coût* des équations de détection de collisions en proposant un mouvement intermédiaire spécifique, puis à réduire leur *nombre*, et montré comment la combinaison d'un mouvement intermédiaire arbitraire, de l'arithmétique d'intervalles et de boîtes englobantes orientées permettait de détections des collisions en continu entre modèles polyédriques rigides complexes en temps réel.

Nous avons ensuite cherché à exploiter le *mouvement* des objets et proposé de compléter les hiérarchies englobantes par des *vecteurs de recul*, qui permettent d'accélérer la détection de collisions significativement lorsque les objets sont proches les uns des autres.

Enfin, nous avons montré que la formulation des problèmes dynamiques dans l'espace des *mouvements*, obtenue à partir du principe des moindres contraintes de Gauss, est mathématiquement équivalente à celle traditionnellement établie dans l'espace des *contacts*, mais présente des avantages algorithmiques. Ceci nous a conduit à proposer un modèle de friction dans l'espace des mouvements qui, s'il est différent du modèle de Coulomb, semble convenir aux applications d'infographie et de réalité virtuelle.

Les algorithmes de détection de collisions continue et ceux de dynamique ont été implantés et couplés pour former une librairie C++ de simulation dynamique interactive, CONTACT Toolkit. Plusieurs bases de données industrielles, fournies par Renault et Airbus-EADS, ont ainsi été testées avec succès, sur plusieurs configurations matérielles. De manière générale, les tests nous ont permis de démontrer le confort apporté

par la détection de collisions continue interactive, notamment grâce au réalisme et à la cohérence de la simulation, en particulier en présence de retour d'efforts.

Ainsi, nous sommes convaincus que la détection de collisions continue présente de sérieux avantages dans une simulation dynamique, en contribuant de façon notable au confort de l'utilisateur. En haptique, notamment, le réalisme visuel semble jouer un rôle important dans la qualité de la raideur perçue par l'utilisateur, dans les tests que nous avons effectués.

De nombreux points restent stimulants et, en particulier, nous aimerions dans l'avenir considérer les sujets suivants :

- La notion de mouvement intermédiaire arbitraire apporte une certaine liberté dans le choix du mouvement, mais pose en même temps la question de l'optimalité du mouvement choisi. Faut-il choisir le mouvement en fonction de sa proximité avec le mouvement réel de l'objet ? Faut-il plutôt le choisir en fonction de l'implantation des équations de détection de collisions ? L'optimalité d'un mouvement intermédiaire dépend ainsi sans doute de l'application, selon des critères qui restent à définir.
- Pour résoudre les problèmes dynamiques, nous avons raisonné sur les mouvements des objets à un instant donné. Ainsi, les différentes contraintes linéaires exprimées dans l'espace des mouvements dépendent des grandeurs physiques à un instant donnée. Il est sans doute possible de formuler les contraintes sur le *mouvement* de l'objet durant un intervalle de temps considéré, en utilisant le mouvement intermédiaire arbitraire. Ainsi, les inconnues ne seraient plus les accélérations à un instant donné, mais les paramètres du mouvement intermédiaire choisi. Déterminer un mouvement qui vérifie les contraintes au cours d'un intervalle de temps donné permettrait donc de s'assurer que les contraintes sont respectées au cours du mouvement, et permettrait sans doute d'utiliser de plus grands pas de temps. De manière générale, il serait sans doute utile de développer une meilleure synergie entre les algorithmes de détection de collisions et ceux de dynamique.
- Nous avons proposé des méthodes valables pour les objets polyédriques rigides. Nous aimerions essayer d'étendre ces méthodes aux surfaces de subdivision, de plus en plus répandues en infographie.
- Il nous semble que la détection de collisions entre objets déformables présente en quelques sorte moins de contraintes puisque, au moins pour une surface polyédrique, le mouvement de chaque point peut être établi indépendamment des mouvements des points voisins. Ainsi, Provot [Pro97] a montré (en l'appliquant à la simulation de tissu) qu'il est possible de réduire les équations de détection de collisions continues entre deux surfaces déformables à des polynômes de degré trois en supposant que les trajectoires des points du maillage durant le pas de temps sont des segments de droite parcourus à vitesse constante[1]. Il serait intéressant de considérer le problème de la détection de collisions conti-

[1]Bien sûr, ce type de mouvement n'est pas utilisable pour les objets rigides puisqu'il conduirait à une déformation de l'objet.

nue entre surfaces rigides et déformables (a priori plus difficile en raison des contraintes posées par la rigidité de l'un des deux objets).

– En haptique, nous avons utilisé le modèle simple du couplage virtuel. Cependant, il est bien connu (et cela se vérifie aussi pour nos algorithmes) que la détection de collisions est beaucoup plus coûteuse que la résolution des problèmes dynamiques. Il serait donc sans doute très profitable de séparer les algorithmes dynamiques de ceux de détection de collisions et de les inclure *dans la boucle haptique*, afin d'accélérer le rendu haptique et accroître ainsi la raideur perçue. Ceci pose cependant un problème nouveau lorsque des algorithmes continus sont utilisés.
– Les travaux récents ont montré un intérêt croissant pour l'exploitation du matériel graphique en détection de collisions. A part le cas d'un outil simple mobile, appliqué à la simulation chirurgicale par Lombardo *et al.* [LCN99], aucun algorithme continu n'a, à notre connaissance, été proposé dans ce cadre. De plus, sur le long terme, il est possible d'imaginer l'intégration des algorithmes de simulation dynamique dans les cartes graphiques.
– Dans le même ordre d'idées, il est possible de remarquer que les *mémoires informatiques* sont en fait mal adaptées à une utilisation des hiérarchies englobantes en détection de collisions (et à une utilisation des structures hiérarchiques dans le cas général). En effet, il est bien connu que la vitesse des mémoires des ordinateurs a évolué beaucoup moins vite que celle de leurs processeurs. Par conséquent, elles sont organisées en *niveaux*, *i.e.* sont associées à des mémoires cache, plus rapides. Afin d'accéder rapidement à une partie de la mémoire, il est donc généralement conseillé de *prévenir* la mémoire centrale de la requête, en quelque sorte, pour qu'elle place les données souhaitées en mémoire cache. De plus, il est généralement souhaitable que des données auxquelles on souhaite accéder à des instants proches l'un de l'autre soient également proches en mémoire. Cependant, parce qu'une structure hiérarchique est par définition non linéaire, *alors que le système d'adressage des mémoires l'est* (il suffit d'une seule adresse pour accéder à la mémoire, et non plusieurs), il est très difficile de déterminer un ordre optimal (et même de savoir s'il existe) pour placer les noeuds de la hiérarchie en mémoire. En effet, en raison des positions changeantes des objets au cours du temps, l'ordre dans lequel il faut accéder aux noeuds de la hiérarchie peut varier. De plus, cet ordre dépend de l'algorithme choisi pour parcourir les hiérarchies. Ce problème, qui serait largement résolu si les mémoires pouvaient être adressées dans deux directions à la fois, nous semble particulièrement intéressant.
– Enfin, deux points nous paraissent essentiels pour l'accélération des méthodes de détection de collisions : l'utilisation de structures hiérarchiques d'une part, et l'exploitation de la cohérence temporelle et spatiale dans la simulation d'autre part. Très peu de méthodes, cependant, parviennent à combiner les deux, essentiellement à cause de l'encombrement mémoire requis pour le stockage des informations déduites du pas de temps précédent [PML97, LC98, Got99, Zac00]. Parvenir à une combinaison efficace de ces techniques nous semble être un développement très souhaitable des futures méthodes de détection de collisions.

Annexe A

Description de la librairie CONTACT Toolkit

Dans cette annexe, nous donnons quelques informations sur l'implantation et l'utilisation de CONTACT Toolkit. La section A.1 explique comment les paramètres du vissage correspondant au mouvement de l'objet sont calculés. La section A.2 donne un aperçu de l'utilisation du simulateur.

A.1 Calcul d'un vissage à partir du mouvement de l'objet

Dans notre implantation, le vissage équivalent au mouvement de l'objet est obtenu à partir des vitesses translationnelle (dx, dy, dz) et rotationnelle (wx, wy, wz) (en radians). Essentiellement, la méthode consiste à décomposer la translation en deux termes, l'un colinéaire à l'axe de rotation, et l'autre qui lui est orthogonal. La partie orthogonale de la translation est ensuite absorbée en décalant l'axe de rotation pour obtenir le vissage équivalent. Cette méthode, classique, est implantée dans la fonction suivante.

```
w=sqrt(wx*wx+wy*wy+wz*wz); // w est la norme du vecteur de rotation

if ((w>=-EPSILON)&&(w<=EPSILON)) w=0; // afin d'eviter les erreurs d'arrondi

if (w==0) { // pas de rotation
   s=sqrt(dx*dx+dy*dy+dz*dz); // s est la norme du vecteur de translation
   if ((s>=-EPSILON)&&(s<=EPSILON)) s=0;
   if (s==0) { // pas de mouvement
      u=cVertex(0,0,1);
```

```
      o=cVertex(0,0,0);
   }
   else {
      cReal t=1/s;
      u=cVertex(dx*t,dy*t,dz*t);
      o=cVertex(0,0,0);
   }
} else { // il y a une rotation

   // on calcule le vecteur unitaire directeur de l'axe du vissage

   cReal v(1/w);
   u=cVertex(wx*v,wy*v,wz*v);

   // on decompose la translation suivant u et son orthogonal

   cVertex t(dx,dy,dz);
   s=(t|u); // decomposition suivant u : intensite de la translation
   cVertex n1(t-s*u); // vecteur orthogonal à u

   n1.Round(); // Pour arrondir les composantes du vecteur
   if (n1[0]||n1[1]||n1[2]) {
      n1.Normalize();
      cVertex n1orth=(u^n1);
      cReal n2x=cos(0.5*w);
      cReal n2y=sin(0.5*w);

      o=0.5*(t|n1)*(n1+n2x/n2y*n1orth);

      }
   else o=cVertex(0,0,0);

}

// Le vissage doit correspondre a un mouvement du point de pivot
// de l'objet exprime dans le repere global pivot_global.
// Il faut donc decaler l'axe du vissage :

o=o+pivot_global;
```

A.2 Exemple d'utilisation

Nous décrivons ici un exemple d'application. Dans cet exemple, une théière se déplace lentement vers un lapin. La trajectoire est essentiellement une translation mais,

à chaque pas de temps, la théière à un léger mouvement de rotation. Sa trajectoire est donc une succession de mouvements intermédiaires généraux.

Pour inclure CONTACT Toolkit dans une application, il faut ajouter la ligne suivante :

```
#include "CONTACT.hpp"
```

Aussi, le compilateur doit avoir accès au fichier `CONTACTToolkit.lib`. Cet exemple requiert deux entêtes supplémentaires :

```
#include <stdlib.h>
#include <iostream.h>
```

La première chose à faire est d'initialiser les objets :

```
void main(void) {

    // lecture des objets
    // NB : les hiérarchies englobantes sont lues en même temps

    pcObject object1=new cObject("teapot.cdo",-1);
    pcObject object2=new cObject("bunny.cdo",-1);

    // positions initiales

    cMatrix4 mat(IDENTITY);
    object1->setGlobalMatrix(mat);
    object2->setGlobalMatrix(mat);

    // vitesses initiales

    object1->setTransVelocity(cVertex(0,0,0)); // vitesse translationnelle
    object1->setRotVelocity(cVertex(0,0,0)); // vitesse rotationnelle
    object2->setTransVelocity(cVertex(0,0,0)); // vitesse translationnelle
    object2->setRotVelocity(cVertex(0,0,0)); // vitesse rotationnelle

    // calcul des propriétés physiques
```

```
object1->ComputeMassProperties();
object2->ComputeMassProperties();
```

Ensuite, l'environnement virtuel doit être initialisé, et les objets doivent être inclus dans la scène :

```
pcScene world=new cScene(2,1); // Au plus 2 objets, dont 1 mobile
int id1=world->AddObject(object1); // id1 est l'id du premier objet
int id2=world->AddObject(object2); // id2 est l'id du deuxieme objet

world->setObjectMobility(id1,1); // la theiere est mobile
world->setObjectMobility(id2,0); // le lapin est statique
```

Maintenant la boucle peut démarrer :

```
do {

    // mouvement aleatoire de la theiere

    double dx=0.02;
    double dy=0.0;
    double dz=0.0;
    double wx=0.05*((rand()/(double)RAND_MAX)-0.5);
    double wy=0.05*((rand()/(double)RAND_MAX)-0.5);
    double wz=0.05*((rand()/(double)RAND_MAX)-0.5);
    world->setGlobalForce(id1,dx,dy,dz,wx,wy,wz);
```

La force agissant sur un objet est décrite par six paramètres. La force appliquée au centre de gravité est donnée par `dx`, `dy` et `dz`, et le couple appliqué à l'objet est donné par `wx`, `wy` et `wz`.

Ensuite, la fonction `NextStep` est appelée. Elle se charge de détecter les collisions et de calculer le mouvement contraint de la théière, suivant l'algorithme général donné dans le premier chapitre (*cf* figure 1.1) :

```
    world->NextStep(); // on avance d'un pas de temps
```

```
}
while (1); // et ainsi de suite
```

Dans cet exemple, la théière avance vers le lapin, glisse sur lui et le dépasse.

Bibliographie

[AS90] K. S. Al-Sultan. Nearest Point Problems : Theory and Algorithms. Ph.D. dissertation, The University of Michigan, Ann Arbor, 1990 135

[AP97] M. Anitescu and F. A. Potra. Formulating Dynamic Multi-Rigid-Body Contact Problems with Friction as Solvable Linear Complementarity Problems. Nonlinear Dynam. 14 (1997), no. 3, 231–247. 37

[APS99] M. Anitescu, F. A. Potra and D. E. Stewart. Time-stepping for Three-dimensional Rigid Body Dynamics. Computational Modeling of Contact and Friction. Comput. Methods Appl. Mech. Engrg. 177 (1999), no. 3-4, 183–197. 37, 40, 123, 135

[BWS99] G. Baciu, S. K. Wong, and H. Sun. *RECODE : An Image-Based collision Detection Algorithm.* Journal of Visualization and Computer Animation, Vol. 10, No. 4, 1999 pp. 181-192. 27

[Bar89] D. Baraff. Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies. In Computer Graphics (Proc.SIGGRAPH), volume 23, pages 223-232. ACM, July 1989. 38, 122, 127, 128

[Bar90] D. Baraff. Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulations. In Computer Graphics (Proc.SIGGRAPH), volume 24, pages 19-28. ACM, August 1990. 26, 34, 38

[Bar91] D. Baraff. Coping with Friction for Non-penetrating Rigid Body Simulations. In Computer Graphics (Proc.SIGGRAPH), volume 25, pages 31-40. ACM, July 1991. 37, 38

[Bar93] D. Baraff. Non-penetrating Rigid Body Simulation. State of the Art Reports of EUROGRAPHICS'93, Eurographics Technical Report Series. 36, 70

[Bar94] D. Baraff. Fast Contact Force Computation for Nonpenetrating Rigid Bodies. In SIGGRAPH 94 Conference Proceedings, Annual Conference Series, pp 23-34. ACM SIGGRAPH, Addison Wesley, 1994. 37, 38, 39, 40, 122, 123, 124, 127, 133, 135

[Bar95] D. Baraff. Interactive Simulation of Solid Rigid Bodies. IEEE Computer Graphics and Applications, 15(3), pp 63-75, May 1995. 144

[Bar96] D. Baraff. Linear-time Dynamics using Lagrange Multipliers. In SIGGRAPH 96 Conference Proceedings, Annual Conference Series, pp 137-146. ACM SIGGRAPH, Addison Wesley, 1996. 38, 124, 134

[BM93] D. Baraff and R. Mattikalli. Impending Motion Direction of Contacting Rigid Bodies. Technical report CMU-RI-TR-93-15, Robotics Institute, Carnegie Mellon University, June, 1993. 125

[Bar99] H. Baruh. Analytical Dynamics. WCB McGraw-Hill, 1999. 124, 125

[Bar00] H. Baruh. Another Look at the Fundamental Equations of Dynamics. Journal of the Chinese Society of Mechanical Engineers, tVol. 21, No. 1, 2000, pp. 15-24.

[BB88] R. Barzel and A. H. Barr. A Modeling System Based on Dynamic Constraints. In Proceedings of the 15th annual conference on Computer graphics, 1988, pp 179-188. 38

[Bau72] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. Comp. Meth. in Appl. Mech. and Eng., 1 :1-16, 1972. 146

[BHB99] P. J. Berkelman, R. L. Hollis and D. Baraff. Interaction with a Realtime Dynamic Environment Simulation using a Magnetic Levitation Haptic Interface Device. Proceedings of the 1999 IEEE International Conference on Robotics and Automation, May 1999, Detroit, Michigan. 19

[BIS00] L. Bouguila, M. Ishii and M. Sato : Scaleable SPIDAR : A Haptic Interface For Human-Scale Virtual Environments. Haptic Human-Computer Interaction 2000 : 182-193. 154

[BV93] W. J. Bouma and G. Vaněček, Jr.. Modeling contacts in a physically based simulation. SMA '93 : Proceedings of the Second Symposium on Solid Modeling and Applications, pp 409-418, 1993.

[Bra91] R. M. Brach. Mechanical Impact Dynamics ; Rigid Body collisions. John Wiley and Sons, Inc., 1991.

[BS02] G. Bradshaw and C. O'Sullivan. Sphere-Tree Construction using Dynamic Medial Axis Approximation. In Proceedings of ACM Symposium on Computer Animation 2002. 29, 58

[BS98] M. Buck and E. Schömer. Interactive rigid body manipulation with obstacle contacts in 6th International Conference in Central Europe on Computer Graphics and Visualization, WSCG'98, pp. 49-56. 38

[BK00] H. Bruyninckx and O. Khatib. Gauss' Principle and the Dynamics of Redundant and Constrained Manipulators. In Proceedings of International Conference on Robotics and Automation, pp 2563-2568, April 2000. 125

[Cam89] S. A. Cameron. Efficient intersection tests for objects defined constructively. International Journal of Robotics Research (USA), vol. 8, no. 1, Feb. 1989, pp 3-25. 32

[Cam90] S. A. Cameron.collision detection by four-dimensional intersection testing. IEEE Trans. Robotics and Automation. 6, 3 (June 1990), pp 291-302. 32, 39

[Can86] J. F. Canny. *collision detection for moving polyhedra.* IEEE Trans. Patt. Anal. Mach. Intell. 8,2 (March 1986), pp 200-209. 32, 39

[CH97] D. A. Carlson and J. K. Hodgins. Simulation levels of detail for real-time animation. In Graphics Interface '97, pages 1–8, 1997. Kelowna, BC, Canada, 21-23 May 1997.

[Cha1831] M. Chasles. Note sur les Propriétés Générales du Système de Deux Corps Semblables Entre Eux, Placés d'une Manière Quelquonque Dans l'Espace ; et sur le Déplacement Fini, ou Infiniment Petit d'un Corps Solide Libre. Bulletin des Sciences Mathematiques de Ferussac, XIV, pp. 321-336. 1831. 50

[Cha99] A. Chatterjee. On the realism of complementarity conditions in rigid body collisions. Nonlinear Dynamics 20 :159-168. 37

[Che00] Stephen Chenney. Controllable and Scalable Simulation for Animation. PhD. Thesis. 2000.

[Chu96] K. Chung. An Efficient collision Detection Algorithm for Polytopes in Virtual Environment. Master's thesis. University of Hong Kong. 1996. 96

[CLMP95] J. Cohen, M. Lin, D. Manocha and M. Ponamgi. *I-COLLIDE : an interactive and exact collision detection system for large-scale environments*. In Proceedings of ACM Interactive 3D Graphics Conference, ACM, Monterey, CA, 1995, pp. 189-196. 26, 27

[CSB95] J. Colgate, M. Stanley, and J. Brown. Issues in the haptic display of tool use. In Int. Conf. on Intelligent Robots and Systems, (Pittsburgh), August 1995. 157

[Col84] T. F. Coleman. Large Sparse Numerical Optimization. Lecture Notes in Computer Science. n.165.

[CD68] R.W. Cottle and G.B. Dantzig. Complementary Pivot Theory of Mathematical Programming. Linear algebra and its applications, 1 :103-125, 1968. 38

[CPS92] R.W. Cottle, J.S. Pang, and R.E. Stone. The Linear Complementarity Problem. Academic-Press, Inc., 1992.

[Cun88] P.A. Cundall. Formulation of a three-dimensional distinct element model - Part I. A scheme to represent contacts in a system composed of many polyhedral blocks. International Journal of Rock Mechanics, Mineral Science and Geomechanics, 25, 1988. 37

[DK85a] D. P. Dobkin and D. G. Kirkpatrick. A linear algorithm for determining the separation of convex polyhedra. J. Algorithms, 6 :381– 392, 1985. 25

[DK85b] D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra - a unified approach. In Proc. 17th. International Colloq. Automata Lang. Program. Lectures notes in Computer Sciences 443, pp 400-413, Springer-Verlag 1990. 26

[Duf92] T. Duff. *Interval Arithmetic and Recursive Subdivision for Implicit Functions and Constructive Solid Geometry*. Computer Graphics, 26(2), July 1992, pp. 131-138. 32, 76

[Dye84] M. E. Dyer. Linear algorithms for two and three-variables linear programs. SIAM Journal on Computing, 13 : pp 31-45, 1984. 25

[ES99] J. Eckstein and E. Schoemer. Dynamic collision detection in virtual reality applications. In 7th International Conference in Central Europe on Computer Graphics and Visualization and Interactive Digital Media, WSCG'99, pp. 71-78. 33, 39

[EH95] E. Eich and M. Hanke. Regularization methods for constraint mechanical multibody systems. ZAMM 75 (1995) 761-773.

[FW99] C.-S. Fahn and J.-L. Wang. collision detection among polyhedral objects. Journal of information science and engineering, 15, 769-799, 1999. 34

[Fau97] F. Faure. Interactive solid animation using linearized displacement constraints. 9th Eurographics Workshop on Computer Animation and Simulation, Lisbon, September 1997.

[Fau99] F. Faure. Fast iterative refinement of articulated solid dynamics. IEEE Transactions on Visualization and Computer Graphics, vol.5, no.3 p. 268-76. July-Sept. 1999. 39

[Fea87] R. Featherstone. Robot Dynamics Algorithms. Kluwer, Boston, MA, 1987. 125

[FD98] C. H. Fedrowitz and F. Düber. An algebra for efficiently solving the continuous collision detection problem. In proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. October 1998. 32

[FMM77] Forsythe, G.E., Malcolm, M.A., and Moler, C.B., Computer Methods for Mathematical Computations, Prentice Hall, Inc., Englewood Cliffs, 1977. 34

[Fra01] Didier François. Manipulation d'objet avec gestion des contraintes physiques. Rapport de DESS Communication, Réseau, Image. Septembre 2001. 153

[FTBAB00] B. Fröhlich, H. Tramberend, A. Beers, M. Agrawala and D.Baraff. Physically-Based Manipulation on the Responsive Workbench. Proceedings of the IEEE VR' 2000 Conference. 122, 153

[GASF94] A. Garcia-Alonso, N. Serrano and J. Flaquer. *Solving the collision detection problem*. IEEE Computer Graphic and Applications, 13(3), 36-43 (1994).

[Gauss1829] K. F. Gauss. Uber ein neues allgemeines Grundgesatz der Mechanik. Journal für die Reine und Angewandte Mathematik, 4, pp 232-235, 1829. 124

[Gen98] Franck Génot. Contributions à la modélisation et à la commande des systèmes mécaniques de corps rigides avec contraintes unilatérales. Rapport de thèse de l'institut national polytechnique de Grenoble. 1998 36, 37

[GB98] F. Génot and B. Brogliato. New results on Painlevé paradoxes. Rapport de recherche n° 3366. INRIA. Février 1998. 37

[Gib79] J. W. Gibbs. On the Fundamental Formulae of Dynamics. American Journal of Mathematics, 2, pp 49-64, 1879.

[GJK88] E. G. Gilbert, D. W. Johnson, S. S. Keerthi. A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space. IEEE J. of Robotics and Automation , Vol.4, No. 2, April 1988. 26

[GMW81] P. E. Gill, W. Murray and M. H. Wright. Practical optimization. Academic Press. 1981. 134

[Gle92] M. L. Gleicher. Integrating Constraints and Direct Manipulation. In Proceedings of the 1992 symposium on Interactive 3D graphics, 1992, pp 171-174.

[Gle94] M. L. Gleicher. A Differential Approach to Graphical Interaction. PhD Thesis. 1994.

[Got99] S. Gottschalk. *collision queries using oriented bounding boxes.* PhD Thesis. 1999. 29, 31, 73, 91, 169

[GLM96] S. Gottschalk, M. C. Lin, and D. Manocha. OBB-Tree : A Hierarchical Structure for Rapid Interference Detection. In SIGGRAPH 96 Conference Proceedings, Annual Conference Series. ACM SIGGRAPH, Addison Wesley, August 1996. 26, 29, 31, 32, 73, 79

[GT01] N. I. M. Gould and P. L. Toint. A Quadratic Programming Bibliography. Numerical Analysis Group Internal Report 2000–1 Rutherford Appleton Laboratory, Chilton, England

[GLGT00] A. Gregory, M. Lin, S. Gottschalk and R. Taylor. *Fast and accurate collision detection for haptic interaction using a three degree-of-freedom force-feedback device.* In Computational Geometry : Theory and Applications. Vol. 15(1-3) : 69-89 (2000). 33

[GMELM00] A. Gregory, A. Mascarenhas, S. Ehmann, M. Lin and D. Manocha. *Six degree-of-freedom haptic display of polygonal models.*In Proc. IEEE Visualization, 2000. 36, 165

[GHZ99] L. J. Guibas, D. Hsu, L. Zhang. H-Walk : Hierarchical distance computation for moving convex bodies. In Proc. ACM Symposium on Computational Geometry, pages 265-273, 1999.

[Han91] M. Hanke. Index reduction and regularization for Euler-Lagrange equations for constraint mechanical systems. In Proc. 2nd Int. Symp. on Implicit and Robust Systems, Warsaw (Poland), July 17-19, 1991, 92-96. 26

[Hac93] W. Hackbusch. Iterative Solution of Large Sparse Systems of Equations. Applied Mathematical Sciences, Vol. 95. Springer-Verlag, New York, U.S.A., 1993.

[Hah88] J. K. Hahn. Realistic animation of rigid bodies. ACM SIGGRAPH Computer Graphics , Proceedings of the 15th annual conference on Computer graphics and interactive techniques June 1988. Volume 22 Issue 4.

[Hel97] Martin Held. ERIT : A collection of efficient and reliable intersection tests. Journal of Graphics Tools, 2(4) :25-44, 1997.

[HKLSSW99] G. Hotz, A. Kerzmann, C. Lennerz, R. Schmid, E. Schömer and T. Warken. Calculation of Contact Forces. In Proceedings of the ACM symposium on Virtual reality software and technology, 1999, pp 180-181. 22

[He99] Taosong He. Fast collision Detection Using QuOSPO Trees. 1999 ACM Symposium on Interactive 3D Graphics, Atlanta, GA, April 1999.

[Hub95] P. M. Hubbard. *collision detection for interactive graphics applications.* Ph.D. Thesis, April 1995. 29

[Hub96] P. M. Hubbard. Approximating Polyhedra with Spheres for Time-Critical collision Detection. ACM Transactions on Graphics, 15(3), July 1996. 29, 57, 58

[JTT01] P. Jiménez, F. Thomas and C. Torras. 3D collision detection : a survey. Computers and Graphics, 25 (2), pp. 269-285, Elsevier Science.

[JDL98] A. Joukhadar, A. Deguet et C. Laugier. A collision model for rigid and deformable bodies. In Proceedings of IEEE ICRA'98. Vol 2. pp 982-988. 25

[Kaw98] K. Kawachi, H. Suzuki and F. Kimura. Technical Issues on Simulating Impulse and Friction in Three Dimensional Rigid Body Dynamics. Computer Animation '98, (June 1998, Philadelphia, Pennsylvania, USA). IEEE Computer Society. 38

[Kea96] R. B. Kearfott. Interval Computations : Introduction, Uses, and Resources, Euromath Bulletin 2 (1), pp. 95-112 (1996). 38

[KZ90] M. McKenna and D. Zeltzer. Dynamic simulation of autonomous legged locomotion. In Computer Graphics (Proc. SIGGRAPH), volume 24, pages 29-38. ACM, August 1990. 77

[KOLM02] Young J. Kim, Miguel A. Otaduy, Ming C. Lin, Dinesh Manocha. Fast Penetration Depth Computation for Physically-based Animation. ACM Symposium on Computer Animation, July 21-22, 2002. 37

[KYK98] Y. Kitamura, A. Yee and F. Kishino. A Sophisticated Manipulation Aid in a Virtual Environment Using Dynamic Constraints Among Object Faces. Presence, vol 7 n 5, october 1998, pp 460-477. 27, 34

[Klo98] James T. Klosowski. Efficient collision detection for interactive 3D graphics and virtual environments. PhD Thesis. State University of New York at Stony Brook. May 1998.

[KHMSZ98] 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 Visualization and Computer Graphics, March 1998, Volume 4, Number 1. 30

[KPLM98] S. Krishnan, A. Pattekar, M. Lin and D. Manocha. *Spherical Shell : A Higher Order Bounding Volume for Fast Proximity Queries. Appeared in Proceedings of WAFR'98.* 26, 29, 79

[KMGL99] S. Kumar, D. Manocha, W. Garrett, and M. Lin. Hierarchical Backface Computation. Computer and Graphics, vol. 9, no. 5, pp. 681-692. Special Issue on Visibility, 1999. 29

[Lec01] Anatole Lécuyer. Contribution à l'étude des retours haptique et pseudo-haptique et de leur impact sur les simulations d'opérations de montage/démontage en aéronautique. Thèse de doctorat. Université Paris XI Orsay. 2001. 106

[LMBLCCG02] A. Lécuyer, C. Megard, J.M. Burkhardt, T. Lim, S. Coquillart, P. Coiffet and L. Graux. The Effect of Haptic, Visual and Auditory Feedback on an Insertion Task on a 2-Screen Workbench. Immersive Projection Technology (IPT) Symposium, Orlando, US, 2002. 154

[LKCGC01] Anatole Lécuyer,Abderrahmane Kheddar, Sabine Coquillart, Ludovic Graux, Philippe Coiffet. A Haptic Prototype for the Simulations of Aeronautics Mounting/Unmounting Operations. IEEE ROMAN'2001, Bordeaux-Paris, France, 18-21.09.2001. 154, 155

[LSW99] C. Lennerz, E. Schömer and T. Warken. A Framework for collision Detection and Response. In 11th European Simulation Symposium, ESS'99, pp. 309-314. 154

[LC98] Tsai-Yen Li, and Jin-Shin Chen. Incremental 3d collision detection with hierarchical data structures. In Proc. VRST'98, pages 139-144. ACM, Taipei, Taiwan, November 1998. 83, 88. 33, 39

[LL82] L. Lilov and M. Lorer. Dynamic Analysis of Multi-rigid Body System Based on the Gauss Principle. Z. Angew. Math. Mechanik, 62(10), pp 539-545, 1982. 26, 169

[Lin93] Ming C. Lin. Efficient collision Detection for Animation and Robotics, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1993. 125

[Lin97] M. C. Lin : Fast and Accurate collision Detection for Virtual Environments. Scientific Visualization 1997 : 171-180. 26

[LC91] M. C. Lin and J. Canny. Efficient algorithms for incremental distance computation. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1008-1014, May 1991. 25, 31

[LC94] M. Lin and J. Canny. A fast algorithm for incremental distance computation. In Proceedings of IEEE International Conference on Robotics and Automation, pp 1008-1014, Avril 1991. 26

[LGEGT99] M. C. Lin, A. Gregory, S. Ehmann, S. Gottschalk, and R. Taylor. *Contact Determination for Real-Time Haptic Interaction in 3D Modeling, Editing and Painting*. Proc. 1999 Workshop for PhanTom User Group. 26

[LG98] M. C. Lin and S. Gottschalk. collision detection between geometric models : a survey. In IMA Conference on Mathematics of Surfaces (San Diego (CA), May 1998), vol. 1, pp. 602– 608. 91

[LCN99] J.-C. Lombardo, M.-P. Cani and Fabrice Neyret. Real-time collision Detection for Virtual Surgery. Computer Animation'99 May 1999. 25

[Loo87] C. Loop. Smooth Subdivision Surfaces Based on Triangles. Masters Thesis, University of Utah, 1987. 28, 169

[LS01] A. Loock and E. Schömer. A Virtual Environment for Interactive Assembly Simulation : From Rigid Bodies to Deformable Cables. In 5th World Multiconference on Systemics, Cybernetics and Informatics (SCI'01), Vol. 3 (Virtual Engineering and Emergent Computing), pp. 325-332.

[Lot81] P. Lötstedt. Coulomb Friction in Two-dimensional Rigid Body Systems. Zeitschrift für Angewandte Mathematik und Mechanik, vol 61. pp.605-615, 1981.

[Lot84] P. Lötstedt. Numerical simulation of time-dependent contact friction problems in rigid body mechanics. SIAM Journal of Scientific Statistical Computing, vol. 5, no. 2, pp. 370- 393, 1984. 36

[Mac60] W. D. MacMillan. Dynamics of rigid bodies. Dover, New-York, 1960. 36, 37, 124

[MRFVVT96] W. R. Mark , S. C. Randolph , M. Finch , J. M. Van Verth , R. M. Taylor. Adding force feedback to graphics systems. Proceedings of the 23rd annual conference on Computer graphics August 1996. 66

[MS94] T. H. Massie and K. Salisbury. The PHANToM haptic interface : A device for probing virtual objects. Proceedings of the ASME Winter Annual Meeting, Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Chicago, IL.

154

[Meg83] N. Meggido. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. SIAM Journal on Computing, 12 : pp 759-776, 1983. 25

[MS01] V. J. Milenkovic and H.Schmidl. Optimization Based Animation. SIGGRAPH 2001 39, 124, 125

[Mir96] B. Mirtich. Impulse-based dynamic simulation of rigid body systems. PhD Thesis. Fall 1996. 33, 39, 87, 96

[MC95] B. Mirtich and J. Canny. Impulse-based Simulation of Rigid Bodies. In Proceedings of Symposium on Interactive 3D Graphics, 1995. 33, 39, 129

[Mol97] Tomas Möller. A fast triangle-triangle intersection test. Journal of Graphics Tools, 2(2) :25-30, 1997. 22

[Moo62] R. E. Moore. Interval analysis and automatic error analysis in digital computation. PhD Thesis, Stanford University, October 1962. 77

[Moo87] P.M. Moore. A flexible object animation system. Master's thesis, University of California, Santa Cruz, 1987. 37

[MW88] M. Moore and J. Wilhelms. collision Detection and Response for Computer Animation. In Computers Graphics (Proceedings of SIGGRAPH 88), Annual Conference Series, pp 289-298. ACM SIGGRAPH, August 1988. 34, 37, 122, 129

[Mor63] J.J. Moreau. Les liaisons unilatérales et le principe de Gauss. C.R. Acad. Sciences Paris, 256 :871–874, 1963. 125

[MP88] J.J. Moreau and P.D. Panagiotopoulos, (Eds), 1988, "Unilateral contact and dry friction in finite freedom dynamics", Nonsmooths mechanics and applications, CISM Courses and Lectures n. 302, Springer Verlag pp. 1-82. 36

[Mur88] K. G. Murty, Linear Complementarity, Linear and Nonlinear Programming, Helderman-Verlag, 1988. http ://www-personal.engin.umich.edu/ ˜ murty/book/LCPbook/index.html

[NAT90] B. Naylor, J. Amanatides W. and Thibault. Merging BSP trees yield polyhedral modeling results. In Proceedings of ACM SIGGRAPH, 1990, pp. 115-124. 31

[Pai1895] P. Painlevé. Sur les Lois du Frottement de Glissement. C. R. Acad. Sci., 121 :112–115, 1895. 37

[Pal87] R.S. Palmer. Computational Complexity of Motion and Stability of Polygons, PhD Dissertation. Cornell University, January 1987. 12, 122, 123

[PW97] F. Pfeiffer and P. Wolfsteiner. Relative Kinematics of Multibody Contacts. In : Proc. of the Int. Mechanical Engineering Congress and Exposition. Nov, 16th-21th. 1997.

[PB88] J. C. Platt and A. H. Barr. Constraint methods for flexible models. Computer Graphics (Proc. SIGGRAPH), vol. 22, pp. 279-288, 1988. 122

[PML97] M. Ponamgi, D. Manocha and M. Lin. Incremental algorithms for collision detection between solid models. IEEE Transactions on Visualization and Computer Graphics , 1997. 26, 169

[PTVF92] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. R. Flannery. Numerical Recipes in C : The Art of Scientific Computing. Cambridge University Press, Cambridge, second edition, 1992. 36, 111

[Pro97] Xavier Provot. collision and self-collision handling in cloth model dedicated to design garment. Graphics Interface, 177-189. 168

[Qui94] S. Quinlan. Efficient distance computation between non-convex objects. In Proceedings of International Conference on Robotics and Automation, pp 3324-3329, 1994. 29, 57, 58

[RKC00] S. Redon, A. Kheddar and S. Coquillart. An Algebraic Solution to the Problem of collision Detection for Rigid Polyhedral Objects. In Proceedings of IEEE International Conference on Robotics and Automation, pp 3733-3738, April 2000. 40

[RKC01] S. Redon, A. Kheddar and S. Coquillart. CONTACT : arbitrary in-between motions for continuous collision detection. In Proceedings of IEEE ROMAN'2001, Sep. 2001. 40

[RKC02a] S. Redon, A. Kheddar and S. Coquillart. Gauss' least constraint principle and rigid body simulations. In Proceedings of IEEE International Conference on Robotics and Automation, May 2002. 41

[RKC02b] S. Redon, A. Kheddar and S. Coquillart. Fast Continuous collision Detection between Rigid Bodies. In Proceedings of Eurographics 2002. September 2002. 40

[RKC02c] S. Redon, A. Kheddar and S. Coquillart. Hierarchical Back-Face Culling for collision Detection. In proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. September 2002. 40

[RG91] E. Rothberg and A. Gupta. Efficient sparse matrix factorization on high performance workstations-exploiting the memory hierarchy.ACM Transactions on Mathematical Software. Vol. 17 , Issue 3 (1991).pp 313-334.

[RK97] D. Ruspini and O. Khatib. Collision/Contact Models for the Dynamic Simulation of Complex Environments. IEEE/RSJ International Conference on Intelligent Robots and Systems :IROS'97. 38, 40, 123, 127, 129, 133, 135, 142

[RKK97] D. C. Ruspini, K. Kolarov and O. Khatib. The Haptic Display of Complex Graphical Environments. Computer Graphics Proceedings, SIGGRAPH 97 pp 345-52 29, 58, 59

[Sam89] H. Samet. Spatial Data Structures : Quadtree, Octrees and Other Hierarchical Methods, Addison Wesley. 31

[SK92] N. Sancheti and S. Keerthi. Computation of certain measures of complexity between convex polytopes : a complexity viewpoint. In Proceedings of IEEE ICRA'92. 3 : 2508-2518, May 1992. 25

[SM95] A. Sanna and P. Montuschi. Spatial bounding of complex CSG objects. IEEE Proc. Comput. Digit. Tech., vol. 142, no. 6, November 1995, pp 431-439. 32

[SS98a] J. Sauer and E. Schömer. A Constraint-based Approach to Rigid Body Dynamics for Virtual Reality Applications. In Proceedings of the ACM Symposium on Virtual reality software and technology 1998, 1998, pp 153-162. 38, 40, 123, 135

[SF91] M. Shinya and M. Forque. Interference detection through rasterization. The Journal of Visualization and Computer Animation, 2, 131-134 (1991). 27

[Sny92] J. Snyder. Interval analysis for Computer Graphics. Computer Graphics, 26(2),pages 121-130, July 1992. 77, 78, 80

[Sny95] J. Snyder. *An interactive tool for placing curved surfaces without interpenetration.* In Proceedings of ACM SIGGRAPH, pages 209-218, 1995. 33, 34, 76

[SWFCB93] J. Snyder, A. Woodbury, K. Fleischer, B. Currin, and A. Barr, Interval Methods for Multi-point collisions between Time-Dependent Curved Surfaces. Computer Graphics, 27(2), pp. 321-334, Aug. 1993. 33, 39, 76, 80, 83, 96

[Son00] Peng Song, Peter R. Kraus, Vijay Kumar and Pierre Dupont. Analysis of Rigid Body Dynamic Models for Simulation of Systems with Frictional Contacts. Accepted for publication in ASME Journal of Applied Mechanics 2000.

[SBB96] M. A. Srinivasan, G. L. Beauregard and D. L. Brock. The Impact of Visual Information of the Haptic Perception of Stiffness in Virtual Environments. ASME Winter Annual Meeting, November 1996. 165

[Sur92b] [Sur92b]M. C. Surles. An algorithm for linear complexity for interactive, physically-based modelling of large proteins. Computer Graphics, 26(2) :221-230, 1992. Proceedings SIGGRAPH 92.

[Sur92c] M. C. Surles. Techniques for Interactive Manipulation of Graphical Protein Molecules. PhD thesis, University of North Carolina at Chapel Hill, 1992.

[ST96] D. E. Stewart and J. C. Trinkle. An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Inelastic collisions and Coulomb Friction. International Journal of Numerical Methods in Engineering, 39 :2673-2691, 1996. 37, 141

[SLY96] C.J. Su, F.H. Lin and B.P. Yen. An Adaptive Bounding Object Based Algorithm For Efficient and Precise collision Detection Of CSG-Represented Virtual Objects. Proceedings of Symposium on Virtual Reality in Manufacturing Research and Education, 1996.

[TPB87] D. Terzopoulos, J. C. Platt and A. H. Barr. Elastically deformable models. Computer Graphics (Proc. SIGGRAPH), vol. 21, pp. 205-214, 1987. 122

[TPSL95] J.C. Trinkle, J.-S Pang, S. Sudarsky and G. Lo. On Dynamic Multi-rigid-body Contact Problems with Coulomb Friction. Zeitschrift fur Angewandte Mathematik und Mechanik, vol.77, no.4 p. 267-79. Oct. 1995 40, 123, 135

[Tom76] J.A. Tomlin. Robust implementation of Lemke's method for the linear complementarity problem. Technical Report SOL 76-24, Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1976. 38

[UK96] F. E. Udwadia and R. E. Kalaba. Analytical Dynamics : a New Approach. Cambridge University Press, 1996. 124, 125

[VDB98] G. Van den Bergen. *Efficient collision detection of complex deformable models using AABB trees.* Journal of Graphics Tools, 2(4) :1-14, 1997. 29, 31, 90

[Van94] G. Vanecek. Back-Face Culling Applied to collision Detection of Polyhedra. Journal of Visualization and Computer Animation, Vol.5, no.1, pp.55-63, 1994. 40, 95, 96, 97, 118

[VHBZ90] Von Herzen, B., A.H. Barr, and H.R. Zatz, Geometric collisions for Time-Dependent Parametric Surfaces. Computer Graphics, 24(4), August 1990, pp. 39-48. 32, 76

[Wil87] J. Wilhelms. Toward automatic motion control. IEEE Computer Graphics and Applications, 7(4) :11-22, 1987. 37

[Wil76] D. R. Wilhelmsen. A Nearest Point Algorithm for Convex Polyhedral Cones and Applications to Positive Linear Approximations. Mathematics of computation, 30, pp 48-57, 1976. 110, 135

[WB97] A. Witkin and D. Baraff. Physically Based Modeling : Principles and Practice. SIGGRAPH Course Notes. 1997. 19, 20, 36, 38

[Wit77] J. Wittenburg. Dynamics of Systems of Rigid Bodies. Teubner, Stuttgart, Germany, 1977.

[WKA00] S. Wookho , K. Kyunghwan , N. M. Amato. An Interactive Generalized Motion Simulator (GMS) in an Object-Oriented Framework. Proceedings of Computer Animation 2000 (CA'2000), May 2000.

[Zac00] Gabriel Zachmann. Virtual Reality in Assembly Simulation – collision Detection, Simulation Algorithms, and Interaction Techniques. PhD thesis, Darmstadt University of Technology, Summer 2000. Fraunhofer IRB Verlag, ISBN 3-8167-5628-X. 26, 31, 36, 169

[ZR01] Gabriel Zachmann and Alexander Rettig. Natural and Robust Interaction in Virtual Assembly Simulation. Eighth ISPE International Conference on Concurrent Engineering : Research and Applications (ISPE/CE2001), July 2001. 152, 153