

Simulation dynamique et applications robotiques

Ammar Joukhadar

▶ **To cite this version:**

Ammar Joukhadar. Simulation dynamique et applications robotiques. RR-3072, INRIA. 1996. inria-00073620

HAL Id: inria-00073620
https://inria.hal.science/inria-00073620

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

SIMULATION DYNAMIQUE ET APPLICATIONS ROBOTIQUES

Ammar Joukhadar

N° 3072

decembre 1996

THÈME 3



ISSN 0249-6399

INRIA
RHÔNE-ALPES

SIMULATION DYNAMIQUE ET APPLICATIONS ROBOTIQUES

Ammar Joukhadar*

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet SHARP

Rapport de recherche n° 3072 — decembre 1996 — 140 pages

Abstract: We describe models and algorithms designed to produce efficient and physically consistent dynamic simulations. These models and algorithms have been implemented within the *Robot*Φ system[55] which can potentially be configured for a large variety of intervention-style tasks such as dextrous manipulations with a robot hand; manipulation of non-rigid objects; tele-programming of the motions of an all-terrain vehicle; and some robot assisted surgery tasks (e.g. positioning of an artificial ligament in knee surgery). The approach uses a novel physically based modeling technique to produce dynamic simulations which are both efficient and consistent according to the laws of the Physics. The main advances over previous works in Robotics and Computer Graphics fields are twofold: the development of a unique framework for simultaneously processing motions, deformations, and physical interactions; and the incorporation of appropriate models and algorithms for obtaining efficient processing times while insuring consistent physical behaviors.

Key-words: physics, dynamics, interaction, contact detection, collision, identification, adaptive time step, adaptive discretization, medical application

(Résumé : tsvp)

* INRIA Rhône Alpes, Equipe SHARP, France

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN (France)

DYNAMIC SIMULATION AND ROBOTICS APPLICATIONS

Résumé : Nous décrivons, dans cette thèse, des modèles et des algorithmes conçus pour produire des simulations dynamiques efficaces et consistantes, dans le contexte de la Robotique d'intervention (c'est-à-dire, pour les tâches robotiques qui impliquent des contraintes fortes sur la nature de l'interaction entre des objets qui ne sont pas forcément rigides). Ces modèles et ces algorithmes ont été intégrés et implantés dans le système *Robot*Φ qui peut être potentiellement reconfiguré pour traiter une grande variété de tâches d'intervention, comme la manipulation dextre d'un objet par une main robotique, la manipulation d'un objet non rigide, la téléprogrammation du mouvement d'un véhicule tout-terrain, ou encore des tâches chirurgicales assistées par robot (par exemple, le positionnement d'un ligament artificiel dans la chirurgie du genou).

L'approche utilise une nouvelle technique de modélisation physique pour produire des simulations dynamiques qui sont à la fois efficaces et consistantes avec les lois de la physique. Les avantages par rapport aux travaux antérieurs dans le domaine de la robotique et de la synthèse d'image sont: le développement d'une structure unique pour traiter simultanément le mouvement, les déformations, et les interactions; et l'incorporation d'algorithmes et de modèles appropriés pour obtenir un temps d'exécution efficace en assurant un comportement consistant avec les lois de la physique. Les contributions principales de ce système sont: l'intégration de la notion du système masse/ressort avec la dynamique d'objets rigides, la discrétisation adaptative basée sur la notion de matrice d'inertie et de centre d'inertie, le pas de temps adaptatif basé sur la notion de l'énergie mécanique pour optimiser le temps de calcul et éviter la divergence numérique, la détection rapide du contact entre polyèdres déformables, et l'identification des paramètres physiques en utilisant les algorithmes génétiques.

Mots-clé : Physique, dynamique, modélisation, mouvement, déformation, interaction, robotique, discrétisation adaptative, pas de temps adaptatif.

Contents

Introduction **6**

1 Simulation Dynamique: Problème, Techniques de Base et Travaux Antérieurs **10**
1.1 Définition du problème 10
1.2 Mouvement et déformation 11
1.2.1 La mécanique des objets rigides 11
1.2.2 Dynamique des objets rigides articulés 13
1.2.3 La méthode des éléments finis 15
1.2.4 Les méthodes globales 17
1.2.5 Le système masses/ressorts 17
1.2.6 Système de particules 19
1.2.7 Conclusion 20
1.3 Détection des collisions 20
1.3.1 Algorithme de Gilbert et al. 21
1.3.2 Algorithme de Lin et Canny 23
1.3.3 Algorithme de Garcia-Alonso et al. 25
1.3.4 Algorithme de Volino & Thalmann 26
1.3.5 Algorithme de Baraff & Witkin 26
1.3.6 Conclusion 28
1.4 Forces de collision 28
1.4.1 Méthode des contraintes 28
1.4.2 Méthode d'impulsion 29
1.4.3 Méthode de pénalité 29
1.4.4 Conclusion 31
1.5 Les simulateurs dynamiques dans les différents domaines 31
1.5.1 La simulation dynamique en synthèse d'image 31
1.5.2 La simulation dynamique en robotique 33
1.5.3 La simulation dynamique en robotique médicale 34
1.6 Conclusion 35

2 Modèle et Modélisation **37**
2.1 Introduction 37
2.2 Description de notre modèle 37
2.2.1 Représentation géométrique 38

2.2.2 Représentation dynamique 40
2.2.3 Connexion 41
2.2.4 Effet des paramètres physique λ et μ sur le comportement d'un objet 43
2.3 La modélisation d'objets 45
2.3.1 Problématique 47
2.3.2 Du modèle géométrique au modèle dynamique 47
2.3.3 La discrétisation automatique et adaptative d'un polyèdre . . 48
2.3.4 Insertion des connecteurs dans le modèle discrétisé 52
2.4 Conclusion 53

3 Mouvement et Déformations **55**
3.1 Introduction 55
3.2 Equation du mouvement 55
3.3 Résolution de l'équation du mouvement 56
3.4 Le pas de temps adaptatif et l'estimation de l'erreur 59
3.4.1 Problème et approche 59
3.4.2 Détermination du pas de temps 60
3.4.3 Traitement de perturbations externes 61
3.4.4 Caractéristiques principales de notre approche 62
3.5 Expérimentations 64
3.6 Conclusion 67

4 Interactions **69**
4.1 Introduction 69
4.2 La détection du contact 70
4.2.1 Description générale de l'approche 71
4.2.2 Elimination des facettes 71
4.2.3 Détermination des paramètres du contact 73
4.2.4 Localisation du contact 79
4.3 Collision 81
4.3.1 Calcul de la force de la collision 81
4.3.2 Calcul de la contrainte de non-interpénétration 84
4.4 Le frottement 86
4.5 La Viscosité 88
4.6 Expérimentations 89
4.7 Conclusion 91

5 La génération du mouvement : Applications robotiques, médicales et graphiques **94**
5.1 Introduction 94
5.2 Le génération du mouvement par une force 95
5.2.1 Objet rigide 95
5.2.2 Objet articulé rigide ou déformable 95
5.3 Le contrôle par des contraintes géométriques 98
5.4 La saisie par une main articulée 98

5.5 Une application médicale: un ligament du genou 104
5.6 Une application graphique: les effets spéciaux sur les images 109
5.7 Conclusion 112

6 Identification **113**
6.1 Introduction 113
6.2 La conservation des propriétés d'inertie 113
6.2.1 Formulation du problème 113
6.2.2 Solution 114
6.3 Identification des paramètres physiques 117
6.3.1 Les algorithmes génétiques: principe de base 118
6.3.2 Application sur le modèle dynamique 119
6.3.3 Expérimentation 123
6.4 Conclusion 129

Conclusion **130**

Introduction

Motivation

La perception, la planification, et les outils de la simulation dynamique, sont les composants essentiels de la Robotique d'intervention, telle que l'entretien de matériel dans un environnement hostile (nucléaire, espace, mer profonde ...), l'inspection de dégât dans des régions contaminées après un accident nucléaire ou chimique, ou l'exploration de nouveaux emplacements (par exemple exploration de l'espace). Les tâches, dans de telles situations, sont extrêmement complexes et impliquent des interactions entre le robot et son environnement ce qui engendrent des forces qui peuvent fortement influencer le comportement du robot (et par conséquent réduire énormément les chances de succès de la tâche).

De telles situations se produisent, par exemple, quand on manipule un objet rigide ou déformable, à l'aide d'une main dextre [57, 56]. Dans ce cas, les déformations des bouts des doigts changent la surface de contact entre la main et l'objet à saisir et augmentent, par conséquent, la force de frottement et la possibilité d'avoir une saisie stable. Sans prendre en compte la variation de la force de frottement, la stabilité sera caractérisée par une configuration géométrique difficile à atteindre avec une bonne précision. La planification du mouvement d'un véhicule tout-terrain sur un terrain accidenté [66, 67], nécessite également la prise en compte de la dynamique du mouvement du véhicule. En fait, certains chemins géométriques peuvent ne pas être valides à cause de phénomènes dynamiques comme le glissement, le dérapage ou la collision. Donc, un chemin libre n'est pas forcément un chemin valide, et il faut vérifier s'il est ou non dynamiquement franchissable. Dans le domaine de la robotique médicale, nous trouvons d'autres applications. L'étude du comportement du ligament croisé au cours du mouvement passif du genou [74, 75], nécessite également un outil de simulation dynamique, car ce ligament est invisible pendant le mouvement du genou.

Problème

Les modèles géométriques classiques développés dans le domaine de CAO-Robotique et le domaine de la synthèse d'image, ne sont évidemment pas adaptés pour traiter de telles interactions. En effet, le but d'un modèle géométrique est de représenter les propriétés spatiales d'un objet (volume, forme, etc.), tandis que la prise en compte des interactions et mouvement d'un objet, dépend de propriétés physiques (masse,

distribution de la masse, rigidité/élasticité, viscosité, force de collision). Le traitement de ces problèmes, nécessite l'utilisation de modèles appropriés (dits modèles dynamiques), capables de représenter à la fois, le mouvement, les déformations, et les interactions entre objets, tout en restant consistants avec les lois de la physique.

Contrairement à ce qui est le cas dans le domaine de la synthèse d'images, le **réalisme** et l'**efficacité** sont les contraintes principales d'un simulateur dynamique que l'on veut utiliser dans le domaine de la robotique. Pour satisfaire le mieux ces contraintes, certains simulateurs dynamiques exigent certaines restrictions (objets rigides mouvement balistique, accélérations limitées, et forces continues [80, 68]) qui sont justifiables dans certains cas mais pas dans le cas général. Bien que la plupart des robots soient rigides, on en trouve quand même avec des parties déformables (comme la main de Salisbury [76]). De plus, les objets manipulés par les robots ne sont pas forcément rigides. Les restrictions d'un mouvement balistique, et la continuité des forces extérieures servent à utiliser des intégrateurs numériques efficaces et à optimiser la détection de la collision entre les différents objets. Dans certains cas ces restrictions ne peuvent pas être satisfaites, ce qui est le cas de la force du contrôle d'un objet au cours d'une opération d'assemblage, qui est une force discontinue car elle dépend du contact qui est un phénomène discontinue.

Le but de notre travail est de construire un simulateur dynamique (le système *Robot*Φ [63, 55, 61, 59, 58, 60, 66, 67]) plus général qui tolère les restrictions précédentes tout en restant *réaliste*, *efficace*, *simple*, et *compatible* avec d'autres modèles utilisés dans le domaine de la robotique.

Contribution

Notre modèle est une intégration de deux modèles, l'un déformable (le système masse/ressort §1.2.5) et l'autre rigide (la mécanique d'objets rigides §1.2.1). Cela rend le simulateur *efficace* et *compatible*. Il est *efficace* car on n'utilise pas un modèle déformable complexe pour modéliser un objet déformable. Il est *compatible* avec les autres approches quand il s'agit d'un objet rigide car il utilise dans ce cas le même modèle dynamique.

Les caractéristiques principales de notre modèle sont les suivantes :

- L'utilisation des polyèdres convexes pour représenter la forme géométrique d'objets, rend notre modèle *compatible* avec les autres modèles géométriques et nous permet de nous servir d'algorithmes efficaces déjà développés pour la détection du contact [69, 39].

- Une approche semi-automatique et adaptative a été développée pour passer de la forme géométrique d'un objet et de sa masse à sa représentation physique tout en conservant ses propriétés d'inertie (matrice et centre d'inertie). Ceci rend la modélisation *simple* et optimise la complexité du modèle tout en restant *consistant* avec le monde réel.

- L'utilisation d'un formalisme numérique explicite pour résoudre l'équation du mouvement, nous permet de prendre en compte les discontinuités résultantes essentiellement de la force de la collision, la force du frottement (passage du

frottement statique au frottement dynamique), et de certains types de force de contrôle. Cela permet aussi de traiter le système comme une boîte noire, dont l'entrée est la force et dont les sorties sont le mouvement et les déformations. Ceci simplifie l'utilisation du système car, dans ce cas, il ne faut pas réécrire les équations du mouvement après chaque modification de l'environnement.

- Le formalisme explicite pose un problème de stabilité numérique que nous avons résolu en développant une approche adaptative, basée sur la notion de conservation de l'énergie mécanique du système. Cette approche nous a permis d'estimer l'erreur commise, d'éviter la divergence numérique et d'améliorer davantage l'efficacité du système.

- Nous avons développé et intégré avec le système, un algorithme qui permet de détecter, en temps linéaire, le contact entre deux polyèdres déformables, et de calculer le volume d'interpénétration entre deux facettes en contact, pour pouvoir calculer d'une manière plus fidèle les amplitudes et les directions des forces de collisions et de frottements. Cet algorithme nous a permis de rendre linéaire la complexité totale du système.

- Nous avons développé également un algorithme d'identification basé sur la notion d'algorithmes génétiques. Cet algorithme permet de trouver les paramètres du modèle (élasticité, plasticité, etc.) pour lesquels le comportement du modèle coïncide avec celui de l'objet réel.

Structure du document

Le mémoire de la thèse est structuré comme suit :

Le premier chapitre représente le problème de la modélisation dynamique, les techniques de base les plus utilisées et les travaux antérieurs dans le domaine.

Le deuxième chapitre représente notre modèle et la manière semi-automatique et adaptative pour modéliser physiquement un objet en se basant sur son modèle géométrique.

Le troisième chapitre montre l'équation du mouvement, sa résolution, et le principe de pas de temps adaptatif qui permet d'estimer l'erreur commise, d'éviter la divergence numérique et de diminuer le temps de calcul.

Le quatrième chapitre explique notre algorithme linéaire pour détecter le contact entre deux polyèdres déformables en mouvement et présente les trois types d'interactions prises en compte dans le modèle: collision, frottement et viscosité du milieu.

Le cinquième chapitre montre comment générer le mouvement d'un objet physique et demontre les capacités du système sur des exemples de nature très différents. La première application concerne l'interaction entre une main articulée et l'objet tenu par celle-ci, la deuxième concerne la génération des effets spéciaux sur les images et la troisième est une application réelle concernant la simulation du comportement dynamique du ligament croisé du genou.

Le dernier chapitre représente l'identification des propriétés d'inertie d'un objet et l'identification des paramètres physiques (élasticité, viscosité, plasticité, facteurs

de la collision, facteurs de frottement) en utilisant et en adaptant les algorithmes génétiques.

Chapter 1

Simulation Dynamique: Problème, Techniques de Base et Travaux Antérieurs

1.1 Définition du problème

La simulation dynamique consiste à simuler le mouvement, la déformation, et l'interaction entre les objets. Les points principaux à résoudre pour construire un tel système sont les suivants :

- Représenter le mouvement d'un objet en fonction des forces extérieures (dynamique directe[1]).
- Représenter l'effet des forces extérieures sur la forme géométrique de l'objet (déformation).
- Détecter les collisions entre les différents objets.
- Calculer la force de la collision entre deux objets.
- Calculer la force de frottement (statique et cinétique) entre deux objets (loi de Coulomb).

Pour chacun des points précédents, il y a une représentation qui convient mieux pour le résoudre. Donc, il faut trouver une représentation qui soit appropriée pour résoudre tous ces points en même temps.

Les simulateurs dynamiques, en général, sont des programmes qui font un nombre d'opérations très important. Pour qu'un simulateur dynamique soit pratiquement utilisable, il faut résoudre les points suivants:

- Optimiser la complexité du système et le temps d'exécution.

[1] Le modèle dynamique direct consiste à calculer le mouvement d'un objet à partir des couples ou des forces. Le modèle dynamique inverse consiste à trouver les couples et les forces à appliquer sur l'objet afin qu'il suive une trajectoire donnée.

- Minimiser l'erreur commise.
- Trouver un moyen de passer d'un objet réel à sa représentation.

La classification des différentes approches peut se faire, soit selon le problème traité (mouvement, déformation, interaction, etc...), soit selon le domaine d'application. Dans la suite nous présentons d'abord certaines techniques de base qui sont le plus souvent utilisées dans les différents systèmes de modélisation dynamique. Ces techniques sont divisées en trois groupes: les techniques qui permettent de calculer le mouvement et/ou les déformations d'un objet quelconque (§1.2), les techniques qui permettent de détecter la collision entre deux objets (§1.3) et celles qui permettent de calculer la force de la collision (§1.4). Puis, nous présentons quelques simulateurs dynamiques dans trois différents domaines : synthèse d'images, robotique et robotique médicale (§1.5).

1.2 Mouvement et déformation

Nous décrivons dans cette section quelques techniques de base qui sont le plus fréquemment utilisées pour calculer le mouvement et/ou la déformation d'un objet quelconque. Nous présentons ainsi la mécanique des objets rigides (§1.2.1) qui permet de calculer le mouvement d'un objet parfaitement rigide, deux formalismes qui permettent de calculer le mouvement d'une chaîne articulée (un cas intermédiaire entre le rigide et le déformable §1.2.2), la méthode des éléments finis (§1.2.3) qui permet de calculer la déformation d'un objet, les méthodes globales qui caractérisent le mouvement et la déformations d'un objet par un système d'équation (§1.2.4), le système masses/ressorts (§1.2.5) qui permet de calculer à la fois le mouvement et les déformations et enfin le système de particules (§1.2.6) qui est une généralisation du système masses/ressorts.

1.2.1 La mécanique des objets rigides

Cette approche, développée il y a très longtemps dans le domaine de la mécanique, s'intéresse seulement au mouvement des objets rigides non articulés (un point matériel peut être considéré comme un cas particulier d'un objet rigide, où l'objet ne peut pas tourner sur lui même). La position d'un objet parfaitement rigide est représentée par la position de son centre d'inertie et son orientation dans l'espace. Ses moments autour des différents axes sont représentés par sa matrice d'inertie I. Les principales caractéristiques de cette approche sont les suivantes [42]:

- Le mouvement d'une masse ponctuelle m est caractérisé par la deuxième loi de Newton:

$$\vec{F} = m\,\vec{\gamma} \tag{1.1}$$

où, $\vec{F}$ est la somme des forces externes appliquées sur cette masse, et γ son accélération. Comme l'accélération est la dérivée de la vitesse ($\vec{\gamma} = \frac{\delta V}{\delta t}$) et la vitesse est la dérivée de la position ($\vec{V} = \frac{\delta P}{\delta t}$), l'équation 1.1 nous permet de calculer la vitesse et la position de cette masse en fonction du temps.

Par exemple, dans le cas où la force $\vec{F}$ est constante nous pouvons intégrer l'équation 1.1, une fois pour obtenir la vitesse $\vec{V}$, et deux fois pour obtenir la position $\vec{P}$:

$$\vec{V} = \vec{\gamma}.t + \vec{V_0} = \frac{\vec{F}}{m}t + \vec{V_0}$$

$$\vec{P} = \frac{1}{2}\vec{\gamma}.t^2 + \vec{V_0}.t + P_0 = \frac{\vec{F}}{2m}.t^2 + \vec{V_0}.t + P_0$$

- Le centre d'inertie d'un objet se comporte comme une masse ponctuelle, dont la masse est égale à celle de l'objet en question, et qui subit les mêmes forces que celles appliquées sur cet objet.

- La rotation d'un objet rigide est caractérisée par l'équation différentielle suivante :

$$\sum \vec{N} = I\dot{\vec{W}} + \vec{W} \wedge (I.\vec{W}) \quad \text{(Equation d'Euler)} \tag{1.2}$$

où $\vec{N}$ est la somme des moments appliqués sur l'objet par rapport à son centre d'inertie G, I sa matrice d'inertie calculée dans un repère centré en G, et $\vec{W}$ la vitesse de rotation de l'objet autour de son centre d'inertie.

L'intégration de cette équation donne la vitesse de rotation de l'objet $\vec{W}$. Par exemple, dans le cas où $\vec{N}$ est constant, on peut écrire que $\dot{\vec{W}} = \frac{\vec{W_t} - \vec{W_0}}{t}$, d'où :

$$\vec{W_t} = \vec{W_0} + I^{-1}(\vec{N} - \vec{W_0} \wedge (I.\vec{W_0}))\ \ t$$

- La somme des forces internes d'un système physique, constitué d'un objet ou de plusieurs objets, est toujours nulle. Cela vient du principe de l'action/réaction. Donc, lorsque deux objets entrent en collision, la force agissante sur l'un est l'inverse de celle agissante sur l'autre. On sait aussi que la trajectoire du centre d'inertie d'un objet ne peut pas être modifiée par l'effet des forces internes.

- Un objet est dit libre si la somme des forces appliquées sur lui est nulle. La vitesse d'un objet libre est toujours constante (pas de dissipation d'énergie). Mais les vitesses relatives des différentes parties d'un système dynamique (qui peut comporter plusieurs objets rigides) peuvent s'annuler à cause des forces internes (les forces de collision entre les différentes composantes).

- L'énergie cinétique E_c d'une masse ponctuelle m dont la vitesse est $\vec{V}$, est donnée par :

$$E_c = \frac{1}{2}m|\vec{V}|^2$$

- La variation de l'énergie potentielle E_p d'une masse ponctuelle m, dont la position est $\vec{P}$ et qui subit une force extérieure $\vec{F}$, est égale au travail de cette force :

$$\Delta E_p = -\int_P^{P+\Delta P} \vec{F} \times \delta\vec{P}$$

- L'énergie mécanique E_m d'une masse ponctuelle m, est la somme de son énergie cinétique et de son énergie potentielle: $E_m = E_k + E_p$. L'énergie mécanique d'un système physique libre est constante: $\frac{\delta E_m}{\delta t} = 0$.

Les avantages de la mécanique des objets rigides sont, sa simplicité, sa rapidité et sa précision. En fait, le comportement d'un objet est caractérisé par deux équations différentielles simples et faciles à résoudre (1.1 et 1.2). De plus, ces deux équations ne contiennent que des paramètres dont les valeurs sont connues (comme la masse, la matrice d'inertie, les forces et les moments).

1.2.2 Dynamique des objets rigides articulés

Deux grandes approches existent pour calculer les équations dynamiques d'une chaîne de corps rigides articulés. Celle de Newton-Euler, qui décrit le comportement dynamique d'un système en termes de forces et moments, et celle de Lagrange qui le décrit en termes de travail et d'énergie [86].

Formalisme de Newton-Euler

Les équations de Newton-Euler font appel aux lois fondamentales de la dynamique des solides (1.1 et 1.2). Elles conduisent à écrire:

$$F_i = m_i \dot{v_i} \text{ (Deuxième loi de Newton)}$$

$$N_i = I_i \dot{w}_i + w_i \wedge (I_i . w_i) \text{ (Equation d'Euler)}$$

où F_i est le résultat des forces appliquées sur le corps C_i de la chaîne cinématique du manipulateur, m_i est la masse de ce corps, v_i est la vitesse du centre d'inertie de ce corps par rapport au repère absolu fixe, N_i est la résultante des couples appliqués sur le corps par rapport à son centre d'inertie, w_i est la vitesse de rotation de C_i par rapport au centre d'inertie et I_i est sa matrice d'inertie calculée dans un repère dont l'origine est le centre d'inertie de C_i.

Ces équations représentent l'effet des forces et couples dûes à la pesanteur ainsi qu'à l'action des corps C_{i-1} et C_{i+1} sur C_i (figure 1.1).

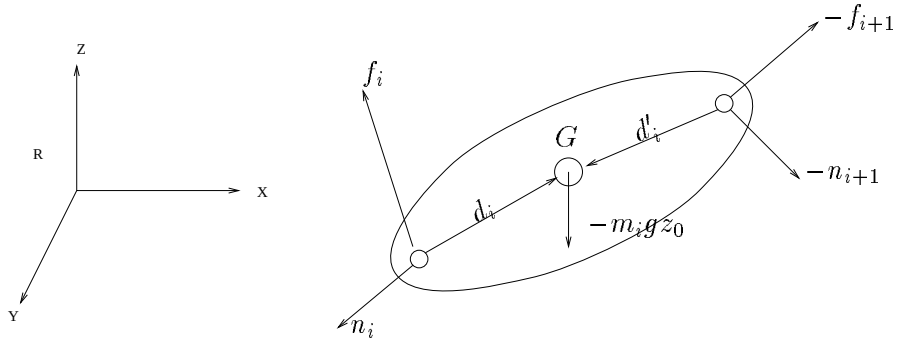


Figure 1.1: *Sur chaque corps C_i de la chaîne articulée, on considère l'effet de la pesanteur, et les actions des corps voisins C_{i-1} et C_{i+1}.*

Donc,

$$F_i = f_i - f_{i+1} - m_i g z_0$$

$$N_i = n_i - n_{i+1} - d_i \wedge f_i + d'_i \wedge f_{i+1}$$

ce qui donne:

$$f_i = f_{i+1} + m_i(\dot{v}_i + g z_0)$$

$$n_i = n_{i+1} + d_i \wedge f_i - d'_i \wedge f_{i+1} + I_i \dot{w}_i + w_i \wedge (I_i w_i)$$

Le calcul itératif du modèle dynamique consiste en un:

- calcul itératif direct des vitesses et des accélérations généralisées ($\dot{w}_i$ et $\dot{v}_i$ $i = 1, 2, ..n$) en utilisant les conditions initiales $w_0 = \dot{w}_o = 0$ et $\dot{v}_0 = 0$.
- calcul itératif inverse des forces généralisées de liaison (f_i et n_i $i = n, n-1, ..1$) en utilisant les conditions finales $f_{n+1} = f$ et $n_{n+1} = n$ où f et n sont les forces et les couples exercées par l'organe terminal sur l'environnement.

Cette méthode autorise un calcul rapide des forces généralisées et paraît intéressante pour réaliser une commande en temps réel. Mais cette technique de calcul numérique conduit à écrire le modèle dynamique sous une forme implicite, qui nécessite que toutes les équations soient calculées à nouveau pour n'importe quel changement de masse, longueur ou inertie.

Formalisme de Lagrange

Ce formalisme est plus approprié que celui de Newton-Euler pour obtenir une forme explicite du modèle. Il consiste à calculer tout d'abord la fonction de Lagrange Ł qui est la différence entre l'énergie cinétique totale T et l'énergie potentielle U du manipulateur:

$$Ł(q, \dot{q}, t) = T(q, \dot{q}, t) - U(q, t)$$

puis à calculer les n équations de Lagrange:

$$\Gamma_i = \frac{d}{dt}(\frac{\delta Ł}{\delta \dot{q}_i}) - \frac{Ł}{\delta q_i} \mid i = 1...n \tag{1.3}$$

qui expriment la force généralisée Γ_i s'exerçant sur chacune des n articulations d'une chaîne mécanique ne comportant pas de boucle fermée.

Le couple Q_i exercé par la pesanteur sur l'articulation i peut s'écrire [86] :

$$Q_i = -\frac{\delta U}{\delta q_i}$$

où U est l'énergie potentielle de la pesanteur. Si on effectue les remplacements dans 1.2.2 on obtient:

$$\Gamma_i = \frac{d}{dt}(\frac{\delta T}{\delta \dot{q}_i}) - \frac{\delta T}{\delta q_i} - Q_i \mid i = 1...n$$

ou encore compte tenu du fait que le manipulateur possède autant de corps que de degrés de liberté:

$$\Gamma_i = \sum_{k=1}^{n} \frac{d}{dt}(\frac{\delta T_k}{\delta \dot{q}_i}) - \frac{\delta T_k}{\delta q_i} - Q_i \mid i = 1..n$$

Cette dernière équation peut être développée et mise sous la forme matricielle suivante [86]:

$$\Gamma + Q = [A]\ddot{q} + [B]\dot{q}\dot{q} + [C]\dot{q}^2$$

où,

$[A]\ddot{q}$ est le vecteur des forces inertielles

$[B]\dot{q}\dot{q}$ est le vecteur des forces de Coriolis

$[C]\dot{q}^2$ est le vecteur des forces centrifuges.

1.2.3 La méthode des éléments finis

Les logiciels de calcul basés sur la méthode des éléments finis [36] sont largement commercialisés. Cette méthode permet de modéliser des matériaux isotropes ou anisotropes, comportement élastique, élasto-plastique, plastique, déformation, déplacement, etc. Nous présentons par la suite comment cette méthode peut servir à calculer les déformations d'un objet.

Cette méthode caractérise les déplacements d'un ensemble de points d'un objet (figure 1.2) par l'équation suivante:

$$\left[\begin{array}{c|c|c} K_{bb} & & K_{bl} \\ \hline\hline K_{lb} & & K_{ll} \end{array}\right] \left\{\begin{array}{c} V_b = 0 \\ \hline\hline V_l \end{array}\right\} = \left\{\begin{array}{c} F_b = 0 \\ \hline\hline F_l \end{array}\right\} \tag{1.4}$$

où la matrice $[K]$ est la matrice de raideur, $[K_{bb}]$ et $[K_{ll}]$ sont des matrices carrées; $[K_{bl}]$ est une matrice rectangulaire, et $[K_{lb}] = [K_{bl}]^t$. V_l est un vecteur de l éléments, où chaque élément représente le déplacement d'un point libre de l'objet. V_b est un vecteur de b éléments qui représentent les déplacements des points bloqués de l'objet (c'est pourquoi ce vecteur est mis à zéro). F_l est un vecteur de l éléments où chaque élément représente la force appliquée sur un point libre (ces forces sont données). F_b est un vecteur de b éléments, où chaque élément représente la force appliquée sur un point bloqué (ces forces sont initialement nulles). Après les déformations F_b sera donné par la résolution de l'équation 1.4. Ce système conduit à deux sous-systèmes:

$$[K_{bl}]V_l = F_b \tag{1.5}$$

$$[K_{ll}]V_l = F_l \tag{1.6}$$

On résoud d'abord le système 1.6 pour obtenir tous les déplacements V_l, puis on calcule la réaction d'appui F_b induite par l'équation 1.5 (figure 1.2).

La résolution de l'équation 1.6 consiste à inverser la matrice de raideur K_{ll} et nécessite donc un nombre d'opérations quadratique $O(l^2)$, où l est le nombre de degrés de liberté (ou le nombre de points libres dont on veut calculer les déplacements).

L'étape la plus importante est le calcul de la matrice de raideur. Ce calcul n'est pas possible dans le cas général. Pour cela la méthode d'éléments finis exige que

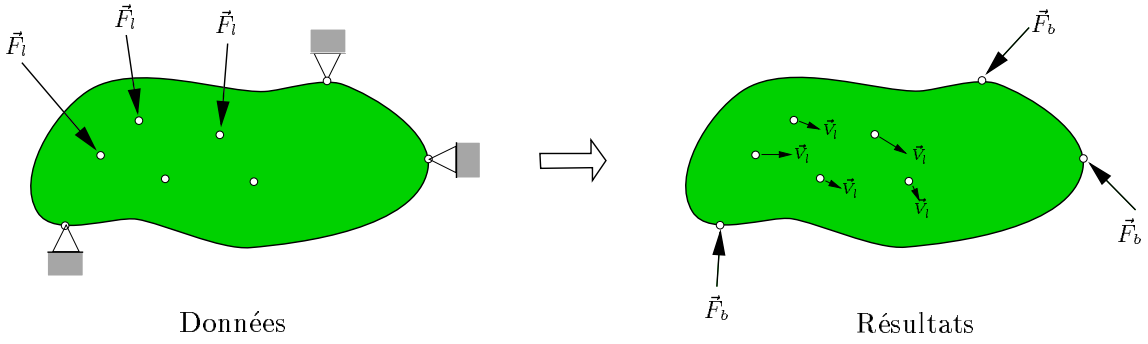


Figure 1.2: *La méthode des éléments finis donne le déplacement de certains points libres d'un objet, en fonction des forces extérieures appliquées sur ces points. Elle donne aussi la réaction d'appui sur les points bloqués.*

l'objet soit divisé en plusieurs éléments réguliers (des cubes ou des tétraèdres) dont on peut calculer les matrices de raideur (figure 1.3). Dans ce cas, le nombre de degrés de liberté sera le nombre de noeuds dans le maillage utilisé. Il faut absolument fixer certains degrés de liberté $b \neq 0$ sinon la matrice de raideur devient singulière et non inversible.

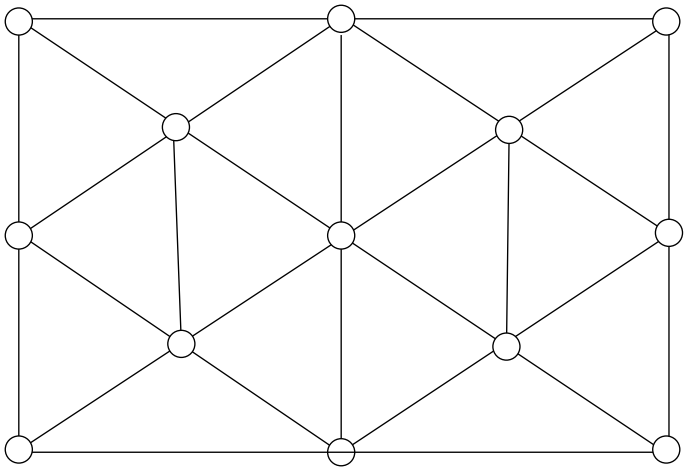


Figure 1.3: *Pour pouvoir calculer automatiquement la matrice de raideur, il faut discrétiser l'objet en plusieurs éléments simples. Le nombre de degrés de liberté dépend du nombre de noeuds dans le maillage utilisé.*

Malgré le réalisme de cette méthode, elle présente certains inconvénients qui le rendent difficile à utiliser dans le domaine de la robotique:

- La précision de la méthode des éléments finis dépend de la finesse de la discrétisation de l'objet initial (le nombre de points libres l considérés). Comme la complexité de cette méthode est d'environ l^2, le temps d'exécution augment significativement en fonction de l. Donc cette méthode ne convient pas les ap-

plications qui demandent la rapidité ou plus particulièrement un temps réel. Cette méthode est utilisée dans certains domaines où la rapidité d'exécution n'est pas une contrainte (le cas de la synthèse d'images [44, 45]).

- L'inversion de la matrice de raideur exige que certains points de l'objet soient fixes. Donc, cette méthode ne peut calculer que les déformations pures de l'objet et non pas son mouvement. Certaines solutions, souvent utilisées dans le domaine de la synthèse d'images [35, 90], consiste à calculer les déformations dans un repère lié à l'objet puis calculer le mouvement de l'objet en utilisant le mécanique d'objets rigides. En utilisant ce type de solutions la méthode des éléments finis perd de son réalisme car le mouvement d'un objet déformable ne peut pas être caractérisé par la dynamique d'objets rigides. En fait le mouvement et la déformation sont liés. Une partie de l'énergie de la force sert à déformer l'objet et une autre partie sert à le déplacer.

1.2.4 Les méthodes globales

Ces méthodes sont basées sur la théorie de l'élasticité. Elles caractérisent le mouvement et les déformations globales d'un objet par une seule équation différentielle [91, 90, 78] :

$$\frac{\delta}{\delta t}(\rho(a)\frac{\delta p(a,t)}{\delta t}) + \mu\frac{\delta p(a,t)}{\delta t} + \frac{\delta \epsilon(p)}{\delta p} = f(p,t) \qquad [78]$$

où,
$p(a,t)$ est la position de la particule a à l'instant t.
$\rho(a)$ est la densité de la masse.
$\mu(a)$ est la densité de la viscosité (facteur d'amortissement à chaque point).
$\epsilon(p)$ est l'énergie potentielle.
$f(p,t)$ est la somme des forces externes appliquées au point p à l'instant t.

La résolution de cette équation nécessite deux types de discrétisations : Une discrétisation spatiale de l'objet en n points, ce qui donne n équations différentielles et une discrétisation temporelle pour résoudre ces n équations. Comme les n équations sont liées l'une à l'autre, la résolution de ce système nécessite un temps de calcul très important.

1.2.5 Le système masses/ressorts

Cette approche a été développée initialement dans le domaine de la mécanique. Elle essaie de simuler la réalité physique de la matière, qui est composée d'un grand nombre de particules interagissant entre elles. Les particules sont représentées par des masses ponctuelles et les forces d'interaction par des ressorts linéaires liant les particules voisines. Ce modèle a donné des bons résultats lors de la simulation du comportement en traction d'une barre [30]. Il a été généralisé par Timoshenko afin de simuler le comportement d'un objet en flexion (Timoshenko-beam [17], figure 1.5). Cette généralisation consiste à remplacer les particules par des plaquettes rigides.

Ce système pose certains problèmes pratiques:

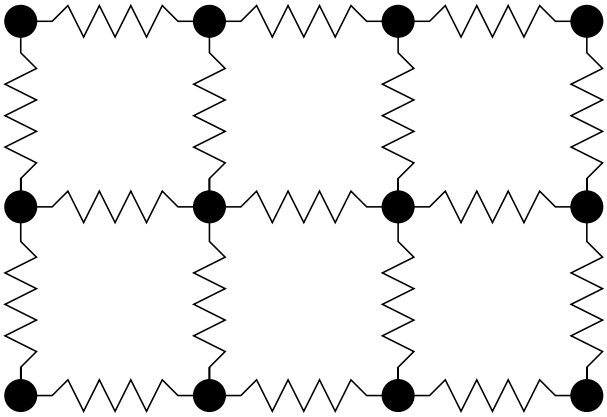


Figure 1.4: *Un objet peut être représenté par un ensemble de particules (masse ponctuelle) liées entre elles par des ressorts/amortisseurs.*

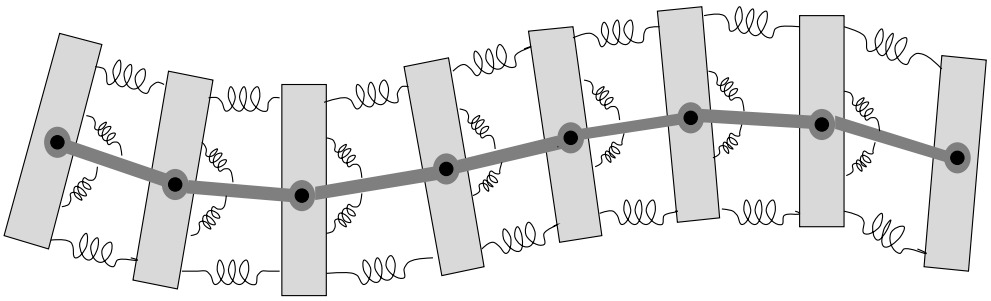


Figure 1.5: *Timoshenko-beam .*

- dans la réalité physique des matériaux, un objet est constitué d'un très grand nombre de particules, ce qui est pratiquement impossible d'être pris en charge par un ordinateur. Par conséquent, le système perd sa généralité quand la simulation nécessite vraiment un tel nombre de particules.

- la résolution de ce système consiste à appliquer la deuxième loi de Newton 1.1 sur chaque particule, en prenant en compte les forces F_i dûes à ses interactions avec les particules voisines et celles F_e dûes à ses interactions avec le milieu extérieur. Donc le système aura la forme suivante :

$$\vec{F_i^k} + \vec{F_e^k} = m^k \, \vec{\gamma^k} \tag{1.7}$$

  La fréquence propre de ce système d'équation 1.7 dépend des valeurs (elle augmente avec la rigidité des ressorts) et de la nature (elle devient infinie si la force est discontinue, ce qui est le cas de la force de la collision) de ces forces.

- Ce modèle donne, comme la plupart des modèles, une représentation paramétrée de l'objet. Donc, un objet peut avoir une infinité de comportement selon les valeurs de ses paramètres (élasticité, viscosité, masse, distribution de la masse). Cela impose la résolution de deux problèmes supplémentaires. D'abord, il faut pouvoir saisir et représenter le comportement de l'objet à modéliser. Puis, il faut pouvoir contrôler le comportement du modèle pour qu'il ressemble à celui de l'objet réel.

Cependant, le système masses/ressorts a intéressé plusieurs travaux de recherche à cause de certains avantages:

- la facilité de mise en oeuvre d'un tel système.

- le modèle reflète la réalité physique des matériaux, ce qui lui permet de simuler une large variété de comportements mécaniques, tels que la traction, la flexion, le mouvement, les déformations élastiques, les déformations plastiques, etc...

- sa complexité peut être linéaire (elle dépend de la méthode de la résolution numérique), et sa complexité en architecture parallèle est constante $O(1)$.

- contrairement à la méthode des éléments finis, ce système permet de calculer en même temps le mouvement et les déformations d'un objet.

Théoriquement ce système doit permettre de calculer, d'une manière simple et efficace, le mouvement et les déformations d'un objet quelconque. Mais il faut d'abord trouver une solution aux problèmes pratiques mentionnés ci-dessus et prolonger le système pour pouvoir calculer les interactions entre les différents objets.

1.2.6 Système de particules

Né dans le domaine de la synthèse d'image, le système de particules est une généralisation du système masses/ressorts. Cette généralisation consiste à donner une dimension géométrique (sphères ou facettes) à ces masses ponctuelles et à ajouter

la possibilité d'une interaction entre les différentes particules qui appartiennent soit au même objet, soit à deux objets différents.

Dans [94] les particules voisine ont été liées par des facettes triangulaires qui peuvent interagir entre elles, afin de simuler les déformations de vêtement. Dans [71] on a donné une orientation à chaque particule pour pouvoir mieux contrôler le comportement de l'objet. Dans [72] et [25], chaque particule a été munie d'une zone de non-pénétration sphérique.

Bien que le système de particule soit une généralisation du système masses/ressorts qui permet de simuler des phénomènes plus complexes (mouvement du sable, fluide, etc..), il est plus compliqué et il nécessite un temps d'exécution plus important. Le fait que la forme géométrique de l'objet soit liée à chaque particule, nécessite un nombre de particules très important pour s'approcher de la forme géométrique de l'objet réel. De plus, cette représentation géométrique n'est pas compatible avec les autres représentations géométriques les plus utilisées et qui sont à base de polyèdres et des facettes.

1.2.7 Conclusion

Il n'y a pas un modèle qui peut être considéré le plus efficace pour calculer le mouvement et les déformations d'un objet. Tout dépend des caractéristiques physiques de l'objet et des contraintes de la tâche à réaliser avec cet objet : pour un objet parfaitement rigide, la mécanique des objets rigides est la plus appropriée : pour un objet rigide articulé, le formalisme de Newton/Euler ou celui de Lagrange sont les plus appropriés : les méthodes des éléments finis et certains méthodes globales, pourtant précis, nécessitent un temps d'exécution très important et elles sont intéressantes pour des applications qui n'exigent pas le temps réel ; le système de particule et celui de masses/ressorts, sont plus rapides que la méthode des éléments finis pour calculer les déformations d'un objet quelconque, mais ils sont des modèles paramétrés qui posent un problème de réglage de ces paramètres.

Les robots sont des objets rigides articulés avec éventuellement des parties déformables (les bout des doigts de la main de Salisbury) et qui manipulent des objets rigides ou déformables. Pour cela et pour que notre modèle soit le plus efficace possible, notre choix consiste à combiner la mécanique des objets rigides avec l'approche de type "masses/ressorts" (§2).

1.3 Détection des collisions

La détection du contact est l'étape la plus critique dans un système de simulation dynamique, car elle nécessite un temps de calcul très important par rapport à celui nécessaire au calcul du mouvement et aux déformations d'un objet.

La détection du contact consiste à calculer la distance entre les deux objets. Quand cette distance est négative, les deux objets sont en contact. La distance est dite négative, si les deux objets s'interpénètrent. Sa valeur est égale à la valeur du déplacement minimal qui permet de séparer les deux objets.

Le temps nécessaire au calcul de la distance entre deux objets, dépend de la forme géométrique de ces deux objets (sphères, cube, polyèdres) et de leurs natures

physiques (rigides ou déformable). Nous présentons dans la suite certaines algorithmes représentatifs, qui sont les plus utiles pour détecter la collision entre deux objets.

1.3.1 Algorithme de Gilbert et al.

Cet algorithme [39] calcule la distance entre les deux enveloppes convexes de deux ensembles de points X et Y. Il donne la valeur de cette distance dans le cas où les enveloppes convexes sont séparées, et il donne une approximation de la distance négative dans le cas contraire.

L'enveloppe convexe d'un ensemble de points X (C_X) est donnée par :
$C_X = \sum_{i=1}^{n} \lambda^i x_i : x_i \in X, \lambda^i \geq 0, \lambda^1 + \lambda^2 + ... + \lambda^n = 1$
La distance euclidienne entre les deux enveloppes convexes est définie par :

$$d(C_X, C_Y) = \min x - y | x \in C_X, y \in C_Y$$

Cette distance est égale à la distance entre la différence de Minkowski et l'origine de repère. La différence de Minkowski $C_A \ominus C_B$ est définie par :

$$C_A \ominus C_B = \{z : z = x - y, x \in C_A, y \in C_B\}$$

donc,

$$d(C_X, C_Y) = \min\{|z| : z \in C_A \ominus C_B\}$$

La différence de Minkowski de deux polyèdres convexes, est lui aussi un polyèdre convexe. Pour calculer la distance de $C_A \ominus C_B$ de l'origine, on introduit la fonction du support h définie sur un ensemble quelconque Z par :

$$h_Z(\eta) = \max z.\eta : z \in Z$$

$z.\eta$ est la projection du point z sur l'axe η, donc la fonction h représente la projection du point le plus loin possible dans la direction η (voir figure 1.6).

Le point z (dite M_Z) de Z le plus proche de l'origine est celui qui est le plus proche dans sa propre direction (ou le plus loin dans l'inverse de sa direction, voir figure 1.6). Donc ce point vérifie la relation suivante :

$$|z^2| + h_Z(-z) = 0$$

Pour trouver le point le plus proche d'un ensemble compact Z à l'origine, on procède de la manière suivante :

1. On commence par un ensemble initial de points $Z_k = z_1, z_2, ..z_m$. Pour $k = 0$, Z_0 est constitué de trois points non alignés si on travaille dans le plan, ou de 4 points non coplanaires si on travaille dans l'espace $3D$.

2. On détermine le point ν_k de Z_k le plus proche de l'origine. Cela peut être fait d'une manière très efficace en utilisant l'algorithme de Johnson [53].

3. Si $|\nu_k^2| + h_Z(-\nu_k) = 0$, alors ν_k est le point de C_Z le plus proche de l'origine et l'algorithme s'arrête.

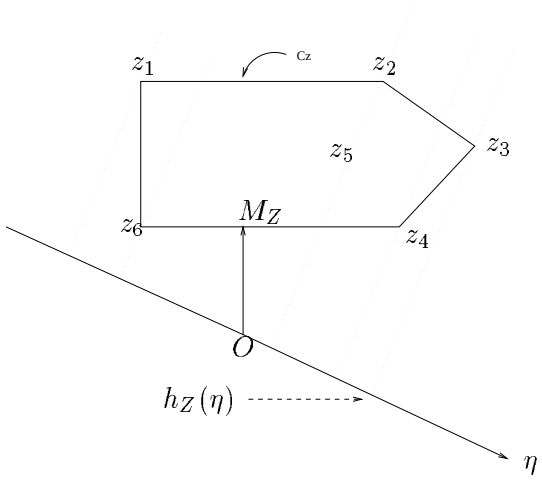


Figure 1.6: *La fonction du support $h_Z(\eta)$ représente la projection du point qui est le plus loin possible dans la direction η. Le point M_Z le plus proche de l'origine est celui qui vérifie que $|M_Z^2| + h_Z(-M_Z) = 0$*

4. Sinon $Z_{k+1} = \hat{Z}_k \cup S_Z$ et on recommence dès l'étape numéro 2, où S_Z est une solution de $h_Z(\nu)$ et $\hat{Z}_k \subset Z_k$ a moins de m éléments et vérifie que $S_Z \in C_{\hat{Z}_k}$.

La figure 1.7 montre un exemple où: $Z = z_1, z_2, z_3, z_4, z_5, z_6$, $Z_0 = z_1, z_2, z_3$, $\hat{Z}_0 = z_2, z_3$, $Z_1 = \hat{Z}_0 \cup z_4$, $\hat{Z}_1 = z_3, z_4$, $Z_2 = \hat{Z}_1 \cup z_5$ et ν_2 est le point le plus proche de Z à l'origine.

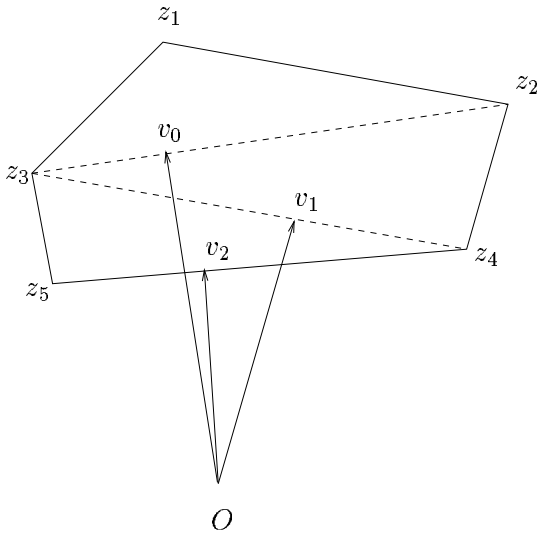


Figure 1.7: *Dans cet exemple l'algorithme de Gilbert converge en trois itérations*

Donc pour calculer la distance entre les enveloppes convexes de deux ensembles de points X et Y, on calcule la distance entre leur différence de Minkowski et l'origine

O en utilisant l'algorithme précédent. L'étape qui prend le plus de temps de calcul dans cet algorithme correspond à l'obtention de $h_{C_A \ominus C_B}$. Cela nécessite, dans le cas général, $n * m$ opérations où n est le nombre de points de A, et m est le nombre de points de B. Une propriété intéressante de la fonction h est que :

$$h_{C_A \ominus C_B}(\eta) = h_{C_A}(\eta) - h_{C_B}(\eta)$$

Donc, en se basant sur cette propriété, le calcul de $h_{C_A \ominus C_B}$ nécessite seulement $n+m$ opérations.

Le nombre d'itérations effectuées par l'algorithme, dépend du choix de Z_0. Au pire des cas, l'algorithme itère $n+m$ fois, ce qui donne une complexité de $O(n+m)^2$. Dans le cas où les deux objets bougent en continue l'un par rapport à l'autre et en utilisant Z_k obtenue à l'itération précédente pour initialiser l'itération actuelle, cette complexité tombe jusqu'à $O(n)$[38].

1.3.2 Algorithme de Lin et Canny

Cet algorithme [69] fournit en temps linéaire les deux points les plus proches de deux polyèdres rigides et convexes. Il peut prendre en compte la continuité du mouvement des deux polyèdres pour rendre la complexité constante. Cet algorithme donne les deux points les plus proches, seulement quand les deux polyèdres sont séparés, sinon l'algorithme ne fonctionne pas.

L'idée de base de cet algorithme consiste à mettre à jour deux caractéristiques (sommet, arête, facette) des deux polyèdres. A chaque itération, on applique certains critères locaux pour vérifier si les deux caractéristiques sont effectivement les plus proches. Si c'est le cas, on s'arrête, sinon on choisit, en fonction de la position relative de ces deux caractéristiques, d'autres caractéristiques qui soient les plus proches les unes des autres. Les critères peuvent être appliqués en temps constant ainsi que le choix des nouvelles caractéristiques. Entre deux itérations successives, la distance entre les deux caractéristiques choisies diminue, donc l'algorithme converge. Le fait que le mouvement des deux polyèdres est continu, fait que les caractéristiques les plus proches changent aussi d'une manière continue. Dans ce cas, à une itération donnée, si les critères ne sont pas vérifiés par les caractéristiques actuelles, ils seront obligatoirement vérifiés par les caractéristiques voisines que l'on peut atteindre en une itération supplémentaire.

Il y a trois critères à satisfaire par une paire de caractéristiques pour qu'elles soient les plus proches: point/sommet, point/arête et point/facette.

1. un point P est le point le plus proche d'un sommet S, s'il se trouve dans une région déterminée par les plans perpendiculaires aux arêtes partantes de S (figure 1.8.a). Si P se trouve à l'extérieur d'un de ces plans, il est donc plus proche de l'arête perpendiculaire à ce plan que de S. Dans ce cas, l'algorithme considère cette arête comme la nouvelle caractéristique et rapplique le deuxième critère pour vérifier si cette arête est la plus proche ou pas de ce point.

2. un point P est le point le plus proche d'une arête A, s'il se trouve dans une région déterminée par quatre plans. Deux plans (Q_a et T_a) sont perpendiculaires à A et passants par ces deux extrémités, et les deux autres plans (L_a

et R_a) sont perpendiculaires respectivement à la facette gauche et à la facette droite de cette arête (figure 1.8.b). Si le point P se trouve à l'extérieur d'un des deux plans (Q_a et T_a) alors il est plus proche du sommet correspondant que de A. Si le point P se trouve à l'extérieur d'un des deux plans (L_a et R_a) alors il est plus proche de la facette correspondante que de A, et l'algorithme va continuer avec la facette correspondante.

3. un point P est le point le plus proche d'une facette F, s'il se trouve dans la région définie par les quatre plans perpendiculaires à F et contenants les arêtes de F (figure 1.8.c) . Si P est à l'extérieur de l'un de ces plans, alors il est plus proche de l'arête correspondante que de F. L'algorithme va donc continuer avec l'arête correspondante.

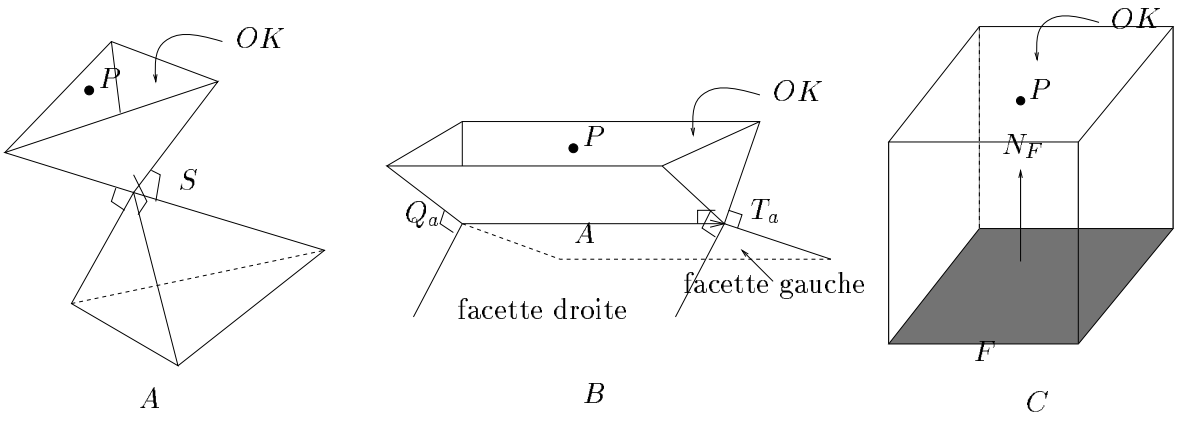


Figure 1.8: *A: Critère point/sommet B: Critère point/arête C: Critère point/facette.*

Ces trois critères permettent de vérifier si deux caractéristiques données sont ou pas les plus proches. Il y a six cas: sommet/sommet, sommet/arête, sommet/facette, arête/arête, arête/facette et facette/facette.

1. Cas sommet/sommet: il faut que chacun d'entre eux vérifie le critère 1 par rapport à l'autre.

2. Cas sommet S/ arête A: il faut que S vérifie le critère 2 par rapport à A et que le point de A le plus proche de S vérifie le critère 1 par rapport à S.

3. Cas d'un sommet S/facette F : il faut que S vérifie le critère 3 par rapport à F et que le point de F le plus proche de S vérifie le critère 1 par rapport à S.

4. Cas arête/arête: il faut que chacun des deux points les plus proches de ces deux arêtes vérifie le critère 1 par rapport à l'autre.

5. Cas arête A/facette F: Si l'arête est parallèle à la facette et qu'elles se recouvrent, dans ce cas elles correspondent aux caractéristiques les plus proches. Sinon l'algorithme recommence en considérant comme caractéristiques A ou un de ses sommets avec une des arêtes de F.

6. Dans le cas de deux facettes, il faut qu'elles soient parallèles et qu'elles se recouvrent. Sinon l'algorithme va retourner la première facette et la plus proche arête de la deuxième puis il recommence.

1.3.3 Algorithme de Garcia-Alonso et al.

L'idée de cet algorithme [33] est de représenter l'objet par plusieurs nivaux de précision. Le premier niveau est un parallèlepipèd dont les facettes sont parallèles aux axes de coordonnées. On appelle ce parallèlepipèd la boîte min-max car sa dimension sur un axe (x, y ou z) est déterminée par les valeurs extrêmes de sa projection sur cet axe ($x_{max} - x_{min}, y_{max} - y_{min}, et \ \ z_{max} - z_{min}$). Le deuxième niveau de représentation est le "container" (voir figure 1.9) qui est le parallèlepipèd minimal qui contient le polyèdre et dont les axes principaux coïncident avec ceux du polyèdre en question. Le troisième niveau de représentation est une matrice $3D$ qui représente la décomposition du container en plusieurs voxels (volume élément). Un voxel est mis à 1 s'il contient une partie de l'objet et à 0 sinon.

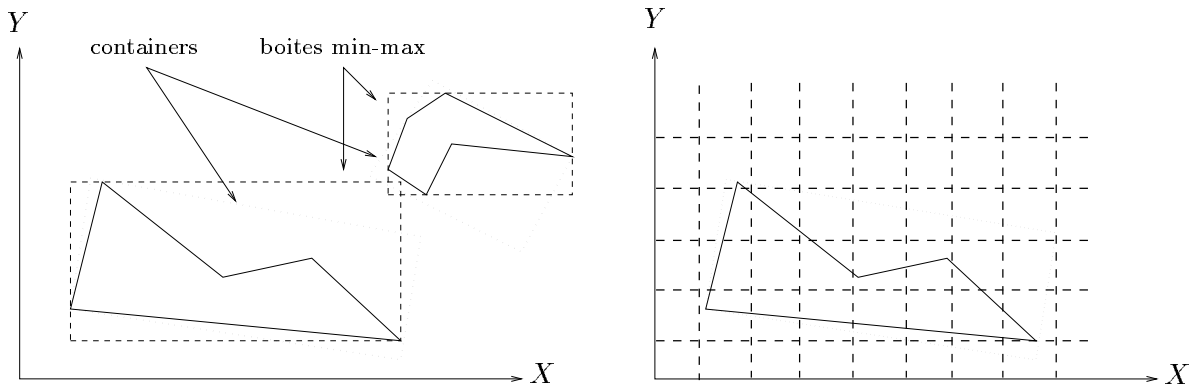


Figure 1.9: *Les boites min-max, les containers et les voxelisations.*

Quand les deux polyèdres sont assez loin l'un de l'autre, un simple test permet de calculer la distance entre leurs boîtes min-max. Quand les boîtes min-max sont en contact, l'algorithme vérifie si les containers sont en contact où pas. Dans le cas où les containers sont aussi en contact, l'algorithme calcule l'intersection de ces deux containers. Cette intersection correspond à deux ensembles de voxels. Pour chaque deux voxels on détecte le contact entre eux et on détermine les facettes qui correspond à ces voxels.

Le container d'un objet, la matrice de voxels et la relation entre les voxels et les facettes de l'objet, peuvent être déterminées statistiquement si les deux objets sont rigides. Quand les objets sont déformables, il faut répéter le calcul à chaque itération, ce qui rend l'algorithme moins efficace.

1.3.4 Algorithme de Volino & Thalmann

Cet algorithme [95] détecte la collision d'un objet surfacique déformable avec soi même. Cette détection est basée sur la courbure de l'objet. Pour que l'objet soit en collision avec lui même il faut qu'il vérifie deux conditions :

- La surface doit être assez courbée pour que deux parties puissent se toucher (figure 1.10.a). Le produit scalaire des deux normales ($N_i.N_j$) aux deux facettes qui se touchent, est forcément négatif. Une surface ne peut pas avoir deux facettes en collision s'il existe un vecteur V tel que $V.N_k > 0$ quelque soit k.
- Le contour de la surface doit être en contact avec lui même (figure 1.10.b). Une surface ne peut pas avoir deux facettes en collision si sa projection selon V ne représente pas une collision avec soit même (figure 1.11).

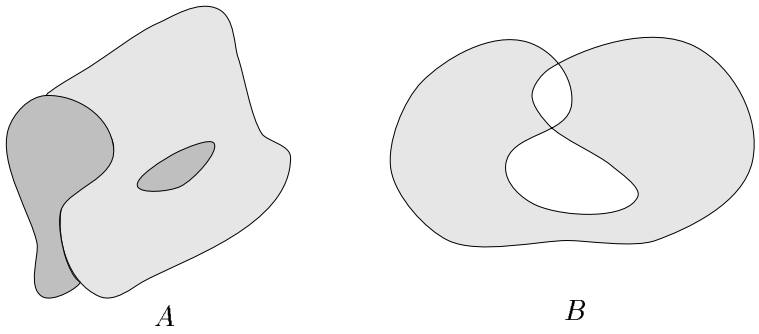


Figure 1.10: *Pour qu'une surface ne soit pas en collision avec lui même, il faut qu'elle ne soit pas très courbée et que sa projection ne soit pas en collision avec lui même.*

Pour trouver le vecteur $\vec{V}$, Volino et Thalmann proposent un algorithme récursif. Ils commencent par une famille initiale de vecteurs qui représente une discrétisation de l'espace. Pour chaque facette on trouve les vecteurs qui donnent un produit scalaire positif avec cette facette et on propage ces informations récursivement (voir figure 1.12). Le temps nécessaire pour trouver $\vec{V}$ est donc de l'ordre de $O(nlog_2 n)$, où n est le cardinal de la famille initiale de vecteurs.

1.3.5 Algorithme de Baraff & Witkin

Baraff [7, 6] a proposé un algorithme capable, en temps linéaire, de détecter le contact entre deux objets rigides. Nous n'allons pas expliquer cet algorithme car il est moins efficace que les algorithmes de Gilbert (§1.3.1) ou Lin (§1.3.2) qui sont largement utilisés dans le domaine de la robotique.

Pour traiter des objets déformables, Baraff & Witkin [9] divisent statiquement chaque objet, en plusieurs sous-objets dont les déformations, pendant le mouvement, sont polynomiales de premier degré. Cela garantie que les sous-objets ne passeront

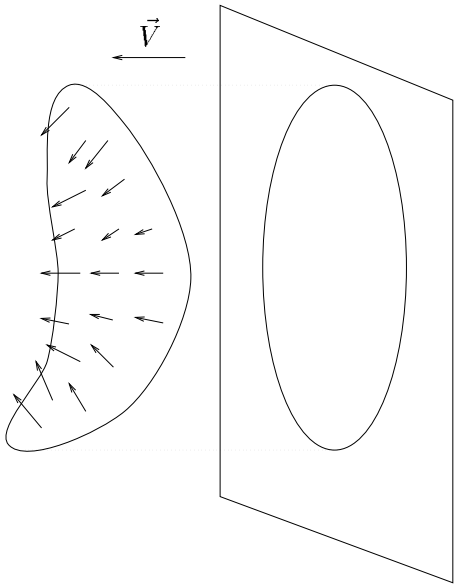


Figure 1.11: *Une surface est considérée non courbée, s'il existe un vecteur $\vec{V}$ dont les produits scalaires avec toutes les normales aux facettes sont positifs.*

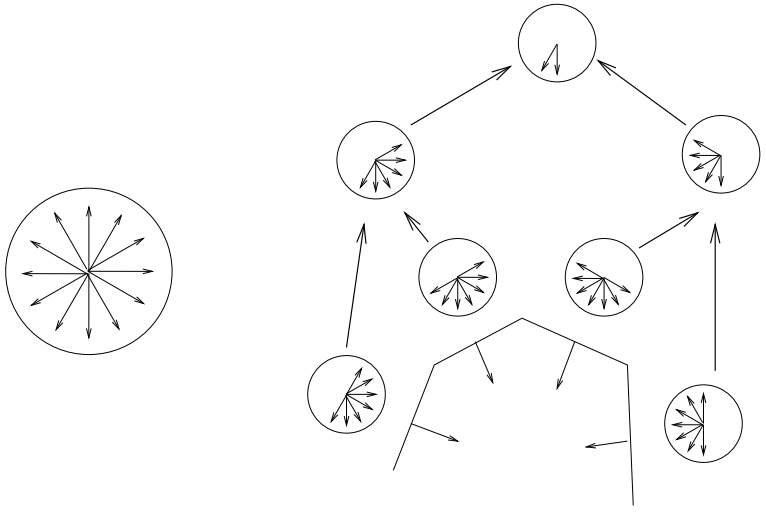


Figure 1.12: *On peut trouver récursivement le vecteur $\vec{V}$, dont les produits scalaires avec toutes les normales aux facettes, sont positives.*

pas à l'état concave lors de leurs déformations. Dans ce cas, on peut utiliser un des algorithmes précédents pour détecter le contact entre deux sous-objets. Le problème provient du fait que le nombre de sous-objets peut être très grand si l'on désire modéliser un objet fortement déformable.

1.3.6 Conclusion

Plusieurs algorithmes sont proposés pour détecter la collision entre des objets en mouvement.

L'algorithme de Gilbert et al. calcule la distance positive entre les deux enveloppes convexes de deux ensembles de points. Il fonctionne même si ces ensembles correspondent aux sommets de polyèdres concaves ou déformables. Dans le cas où les deux ensembles s'interpénètrent, l'algorithme donne une approximation de la distance négative entre eux. La complexité totale de cet algorithme est de $O(n+m)$ où n est le cardinal du premier ensemble et m est celui de l'autre.

L'algorithme de Lin et Canny ne fonctionne que pour des polyèdres convexes rigides et qui ne s'interpénètrent pas. Sa complexité est $O(1)$ quand les deux polyèdres bougent en continue l'un par rapport à l'autre.

L'algorithme de Garcia-Alonso et al. fonctionne pour des objets rigides et déformables mais sa complexité pour des objets déformables n'est pas satisfaisante.

Algorithme de Volino & Thalmann est efficace, mais il fonctionne seulement pour détecter le contact d'une surface avec elle même.

Algorithme de Baraff & Witkin fonctionne pour des objets dont les déformations sont de premier ordre mais il n'est pas efficace pour des objets très déformables.

L'algorithme de Gilbert et al. est le plus approprié pour détecter la collision entre les enveloppes convexes de deux polyèdres déformables en mouvement mais il a besoin d'être adapté afin de détecter la vraie collision entre les deux polyèdres, c'est ce qui a été fait dans le cadre de ce travail (§4.2).

1.4 Forces de collision

Il n'y a pas de solution générale pour calculer la force de la collision entre deux solides. Les différentes solutions peuvent être mise en trois classes : la méthode des contraintes, la méthode basée sur la notion d'impulsion, et la méthode de pénalité.

1.4.1 Méthode des contraintes

La méthode des contraintes consiste à représenter la force de la collision par une contrainte de non-pénétration et à prendre en compte explicitement cette contrainte dans l'équation du mouvement:

$$p(t) = f(force) \tag{1.8}$$

$$C(p(t)) \geq 0 \tag{1.9}$$

L'équation 1.8 caractérise le mouvement de chaque objet en fonction des forces externes, et l'équation 1.9 représente les contraintes de non-pénétration entre ces deux objets.

Baraff [4] a calculé analytiquement le mouvement de chaque objet sous la condition que cet objet est libre (alors sa trajectoire est déterminée seulement par les conditions initiales). Dans ces conditions l'équation du mouvement est continue ce qui permet de l'intégrer efficacement par des approches adaptatives telles que Runge-Kutta ou Adams-Moulton[83, 85]. Quand une collision a lieu, Baraff calcule analytiquement les nouvelles conditions initiales et il relance une nouvelle intégration de l'équation du mouvement.

Le calcul des nouvelles conditions initiales s'appelle la résolution de la collision. La résolution de la collision consiste à résoudre deux équations 1.8 et 1.9 ensemble. Baraff a résolu les contraintes de non-pénétration pour une collision sans frottement [4], il a fait la même chose pour une collision avec frottement mais l'algorithme était NP-complet dure [5, 8]. Cette approche a été adaptée par Baraff et Witkin [9] pour le cas où les objets sont déformables.

Cette méthode est intéressante en absence de frottement et sous les conditions que les collisions ne sont pas très fréquentes et qu'entre deux collisions les objets sont libres.

1.4.2 Méthode d'impulsion

Les méthodes basées sur la notion de la force d'"impulsion" : l'impact [11] est une collision quasi-instantanée (soit de durée négligeable) pendant laquelle les deux objets (parfaitement rigides) exercent une force (dite impulsion) très importante sur eux même. Pour deux masses ponctuelles en contact, la relation entre la vitesse relative de ces deux masses avant l'impact v_i et la vitesse relative après l'impact v_0 est :

$$v_0 = -e\ v_i$$

où e est le coefficient de la restitution. Si $e = 0$ la collision est parfaitement élastique, et si $e = 1$ la collision est parfaitement plastique. e peut être, également calculé par l'hypothèse de Poisson [96, 11]:

$$e = \frac{P_r}{P_c}$$

où P_c est l'impulsion pendant la phase de pression et P_r est cette force pendant la phase de restitution.

Contrairement à la méthode des contraintes, la méthode d'impulsion [80] ne prend pas explicitement les contraintes de non-pénétration dans l'équation du mouvement. Quand les deux objets sont suffisamment proches, l'impulsion est calculée de sorte qu'elle soit capable de les séparer en une seule itération. Dans ce cas aussi on met la condition que les objets entre deux collisions sont en mouvement balistique[2]. Cette condition permet d'optimiser le temps de calcul car le temps de la résolution de la collision n'est pas négligeable.

1.4.3 Méthode de pénalité

La méthode dite de "pénalité": dans le cas général, les objets sont déformables, et la période de la collision ne peut être considérée comme négligeable. Les méthodes dites

[2] Une trajectoire est dite balistique, s'elle ne dépend que des conditions initiales.

de "pénalités" [81] sont alors mieux adaptées pour traiter la collision. Le principe de ces méthodes consiste à générer dynamiquement une force répulsive qui dépend de la valeur de l'interpénétration fictive entre les deux objets. La force engendrée par cette méthode nécessite plusieurs itérations pour séparer les deux objets, à l'opposé des méthodes d'impact qui séparent les deux objets en une seule itération. Le plus souvent la force répulsive est générée par un mécanisme de "ressort/amortisseur" (figure 4.11), et elle peut être donnée dans ce cas par la formule suivante:

$$\vec{F_c} = \begin{cases} (-\lambda x - \mu \dot{x})\vec{k} & \text{si } x < 0 \\ \vec{0} & \text{sinon} \end{cases} \tag{1.10}$$

où λ est la rigidité de la collision, μ la viscosité de la collision (qui représente la dissipation de l'énergie), x est la distance entre les deux points qui sont en contact, $\dot{x}$ est la dérivée de x, et $\vec{k}$ est la direction de la collision (dirigé d'un point vers l'autre).

Moore et Wilhelmes [81] ont utilisé un ressort à deux coefficients de rigidité pour respecter l'hypothèse de Poisson (§1.4.2):

$$k_{restitution} = e k_{compression}$$

Cette méthode a cependant plusieurs inconvénients [73]:

- La force de collision est discontinue. D'après l'équation 1.10 la force de la collision à l'instant t_0^- est 0, par contre sa valeur à l'instant t_0^+ est $-\mu\dot{x}$. Cette discontinuité peut augmenter la fréquence propre de l'équation du mouvement et ralentir, par conséquent, le temps d'exécution.

- Le coefficient de restitution e dépend de la masse des deux objets en collision. Pour une masse m qui entre en collision avec un objet dont la masse est infinie (comme la terre), le coefficient de restitution est [73]:

  $$e = e^{\frac{-\lambda\pi}{\sqrt{4m\mu - \lambda^2}}}$$

  Or le coefficient de restitution est une propriété intrinsèque de la matière (il ne doit donc pas dépendre de la masse). Il est prouvé qu'à faible vitesse d'impact et pour la plupart des matériaux dans le domaine élastique, le coefficient de la restitution peut être approché par [47, 41]:

  $$e = 1 - \alpha v_i \tag{1.11}$$

  où α est une constante.

- La force engendrée par l'amortisseur tend à empêcher les deux objets de se séparer pendant la phase de restitution. Car x tend vers 0 et la vitesse relative à une direction négative (voir l'équation 1.10).

Un modèle non linéaire a été proposé par [47] et étudié par [73] afin de surmonter ces problèmes. Ce modèle consiste à introduire une relation entre la force

d'amortissement $-\mu\dot{x}$ et la distance d'interpénétration x. La force de la collision est alors donnée par:

$$\vec{F_c} = \begin{cases} (-\lambda x^n - \mu \dot{x} x^n)\vec{k} & \text{si } x < 0 \\ \vec{0} & \text{sinon} \end{cases} \tag{1.12}$$

où n est souvent très proche de 1 (elle dépend de la géométrie de la surface de la collision). L'avantage majeur de ce modèle est que le coefficient de restitution ne dépend que de la vitesse relative des deux objets. Si on choisit $\mu = \frac{3}{2}\alpha\lambda$, le coefficient de restitution, pour des faibles valeurs de α, vérifie l'équation 1.11.

1.4.4 Conclusion

Trois méthodes principales permettent de calculer la force de collision entre deux objets. La méthode des contraintes fonctionne pour des objets rigides ou déformables, mais elle nécessite un temps de calcul très important et notamment quand il faut prendre en compte les forces de frottement.

La méthode de l'impulsion est plus efficace mais elle ne fonctionne que pour des objets parfaitement rigides.

La méthode de pénalité permet de calculer la force de collision entre des objets rigides ou déformables. La méthode de pénalité non linéaire permet d'obtenir un comportement plus réaliste et plus efficace car elle évite la discontinuité de la méthode classique.

1.5 Les simulateurs dynamiques dans les différents domaines

Par la suite, nous allons représenter des systèmes de modélisations dynamiques qui utilisent les techniques précédentes classés par type d'applications.

1.5.1 La simulation dynamique en synthèse d'image

La simulation dynamique est un outil indispensable dans le domaine de la synthèse d'image pour produire des animations comportementales, dans lesquelles le comportement d'un objet est automatiquement généré pour ressembler à la réalité.

Terzopoulos et al [91] représentent les déformations globales d'un objet en utilisant une méthode globale (§1.2.4). Cela nécessite un temps de calcul très important. Pour rendre le calcul moins lourd, Terzopoulos et Fleischer [90] ont séparé le mouvement de la déformation. D'abord, ils calculent le mouvement de l'objet en utilisant le mécanique d'objets rigides, puis ils calculent les déformations dans un repère local lié à l'objet. Ce type d'optimisation donne un comportement logique mais il n'est pas correct car la translation d'un objet déformable ne coïncide pas avec celle d'un objet rigide et elle dépend de l'élasticité et de la viscosité de l'objet. Terzopoulose et al.[78] ont ajouté la possibilité de résoudre l'équation précédente sous contraintes sur le mouvement de l'objet.

Carignan et al [18] utilisent la même équation du mouvement que Terzopoulos pour simuler les déformations des vêtements pendant le mouvement. Le calcul, en

utilisant un modèle global, reste très lourd. Pour cela Volino, Courchesne et Thalmann [94] ont utilisé le système de particules (§1.2.6) pour simuler les déformations des vêtements avec plus d'efficacité et sans restrictions sur les conditions de la simulation. Ils utilisent l'algorithme de Volino & Thalmann pour détecter la collision (§1.3.4)

Witkin et Welch [98] utilisent aussi une méthode globale pour calculer les déformations d'un objet, mais ils mettent l'hypothèse de déformations de premier ordre (dans l'espace des contraintes) pour rendre la matrice de contraintes constante. Donc, ils peuvent inverser cette matrice une seule fois à la phase d'initialisation.

Gouret et Thalmann[43] utilise la méthode des éléments finis (§1.2.3) pour simuler les déformations dues au contact entre un objet et une main pendant l'opération de la saisie. La méthode des éléments finis, pourtant correct, mais les calculs à faire sont très lourds.

L'équipe *ACROE* a développé un système de modélisation physique (Cordis-Anima [72]) qui est une généralisation de système masses/ressorts (§1.2.5) où les différentes masses ont une dimension géométrique (représentée par une zone sphérique de non-pénétration) qui leur permet d'interagir entre elles. Ce système permet de modéliser à la fois des objets fluides comme l'eau, le sable [21], etc, et d'intégrer le mouvement avec le son qui en résulte [16, 82]. Pourtant le formalisme de Cordis-Anima permet de simuler une large variété de comportement, mais cela nécessite un temps d'exécution très important vue le nombre de particules nécessaire pour s'approcher de la forme exacte de l'objet et ses caractéristiques physiques.

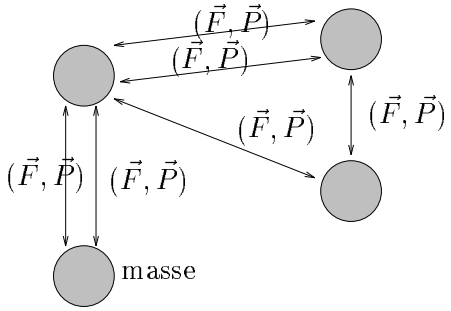


Figure 1.13: *Dans le système de particules, les particules ont des dimensions physiques, qui leur permettent d'interagir dynamiquement. Les forces d'interaction entre les particules sont déterminées en fonction des positions des particules.*

L'équipe *iMAGIS* [3] représente un objet [34, 65, 35] par son squelette rigide et représente sa forme géométrique par une surface implicite [46, 14, 92] générée par une équipotentielle. Donc, ce modèle fait appel à la dynamique d'objets rigides (§1.2.1) pour calculer le mouvement de l'objet et il représente les déformations par un changement de potentiel sur la surface de l'objet. Ce formalisme permet d'optimiser le temps nécessaire pour calculer le mouvement et les déformations. Par contre, il ne permet de simuler que des déformations locales, et par conséquent il ne permet pas de représenter une barre flexible. De plus, le mouvement et les déformations sont

[3] INRIA Rhône Alpes et laboratoire GRAVIR, Grenoble, France

calculés séquentiellement ce qui donne un mouvement incorrect car la dissipation de l'énergie due à la déformation de l'objet n'est pas prise en compte. L'énergie totale de la force est utilisée pour déplacer celui-ci.

Arnaldi et al.[4] ont défini un modèle extensible pour créer des objets hybrides à partir de leurs modèles géométriques, cinématiques et dynamiques [1, 23, 24]. Chaque partie de l'objet est représentée en utilisant le modèle qui convient le mieux (système de particules (§1.2.6), éléments finis (§1.2.3), mécanique d'objets rigides (§1.2.1)). Ce système combine le principe de travail virtuel avec les multiplicateurs de Lagrange ou la méthode de pénalité, afin de générer la forme symbolique de l'équation du mouvement et pour la simplifier. Cela permet de minimiser l'erreur numérique pendant la résolution de cette équation car l'équation même est exacte. Le système est muni d'un langage de modélisation pour faciliter la description d'un objet à partir de son modèle géométrique, ses caractéristiques physiques, les forces dont il subit et les autres types de contraintes.

1.5.2 La simulation dynamique en robotique

Les simulateurs dynamiques sont peu utilisés dans le domaine de la robotique à cause de leur complexité. En fait, l'efficacité est une contrainte essentielle pour la plupart des applications robotiques. L'importance et la nécessité de tels simulateurs pour étudier des interactions complexes entre le robot et son environnement, a poussé certains roboticiens à aborder ce problème.

Paul U. Lee et al[68] ont construit un système de simulation dynamique pour tester les différents contrôleurs et environnement de travail. Ils ont simulé le comportement d'une grande charge lorsque celle-ci est déplacée, afin de valider la faisabilité de certaines stratégies. Ils ont également simulé un environnement comportant deux manipulateurs PUMA. Le but est de tester des applications robotiques en présence de singularité. Le modèle dynamique permet au contrôleur de passer d'une manière lisse par des positions singulières. Ils ont repris la méthode des contraintes de Baraff [4] (§1.4.1) pour calculer la collision entre deux objets et ils l'ont prolongé pour traiter des objets articulés (robot). Ils ont repris également l'algorithme de Gilbert (§1.3.1) pour calculer la distance entre les différents objets [38].

Mirtich and Canny[80] ont développé un simulateur dynamique basé sur la notion d'impulsion (§1.4.2). Ce système ne considère pas explicitement dans l'équation du mouvement, les contraintes de non-pénétration entre les objets. Quand deux objets sont en collision, il calcule la force d'impulsion qui permet de séparer en une seule itération ces deux objets. Le système utilise l'algorithme de Lin and Canny (§1.3.2[69]) pour mesurer la distance entre les différents objets. Il met l'hypothèse que les accélérations sont limitées et qu'un objet entre deux collisions successives est en mouvement balistique afin d'optimiser le temps de calcul.

Les équipes SHARP (INRIA Rhône Alpes) et ACROE ont utilisé le système Cordis-Anima (§1.5.1) pour simuler les interactions d'un véhicule avec le sol. Cette simulation sert à valider les trajectoires géométriques données par un planificateur géométrique afin de trouver une trajectoire qui soit dynamiquement faisable [50, 51, 52]. Le modèle a été optimisé par [22] en introduisant la mécanique d'objets rigides.

[4] IRISA/INRIA Rennes, France

1.5.3 La simulation dynamique en robotique médicale

Les modèles déformables ont aussi des applications dans le domaine médical.

Dans l'équipe Epidaure[5], Delingette et al.[28, 27] ont développé un simulateur dynamique pour simuler les la chirurgie craniofacial. Le but est de prédire les effets d'une modification crânienne sur la forme du visage. Leur système est muni d'une main virtuelle pour modifier le crâne (figure 1.14) et un modèle déformable pour simuler les déformations du visage (figure 1.15).

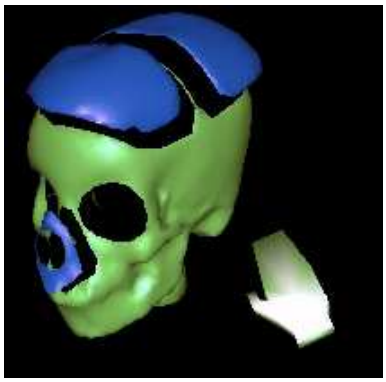


Figure 1.14: *La main virtuelle permet de couper et bouger des parties du crâne [29].*

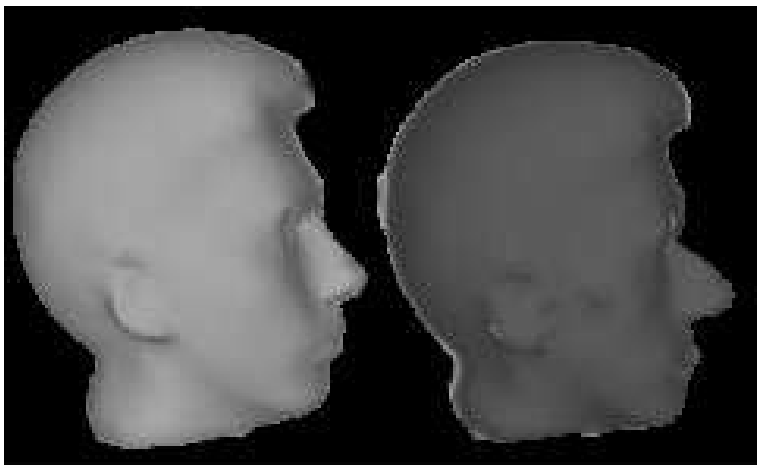


Figure 1.15: *La forme du visage avant et après la simulation de l'intervention [29].*

Dans l'équipe $TIMC$[6], Promayon et al. ont développés un simulateur dynamique [84] pour modéliser et simuler la respiration. Leur but est de simuler la respiration spontanée et artificielle pendant les différentes situations (repos, marche, parole, etc...). Cette simulation peut servir à analyser des situations physiopathologique

[5] INRIA Sophia- Antipolis

[6] Grenoble, France

(éducation/formation, simulateur d'anesthésie, réhabilitation). Ce modèle (dit mémoire de forme) est basé sur la notion du système masses/ressorts (§1.2.5). Dans le cas de la mémoire de forme, les particules ne sont pas liées statiquement par des ressorts. Les ressorts sont dynamiquement ajoutés pour lier chaque particule avec sa position "idéale". Les positions idéales sont calculées d'une manière globale en fonction de la forme actuelle de l'objet et de sa forme au repos. Les particules sont distribuées seulement sur la surface afin de mieux contrôler leurs mouvements et d'ajouter des contraintes particulières sur le comportement d'un objet tel que la contrainte d'un volume constante (car c'est le cas de l'abdomen qui se déforme à volume constant pendant la respiration). Le fait que les particules sont distribuées sur la surface nous oblige de surestimer la matrice d'inertie de l'objet .

Les modèles dynamiques sont très utiles pour la simulation du comportement du ligament croisé du genou, afin de le reconstruire. La simulation du comportement du ligament été le sujet de plusieurs projets de recherche, où on le modélise par des lignes élastiques qui lient le fémur et le tibia [70, 15], par un modèle viscoélastique [99] ou par la méthode des éléments finis (§1.2.3) [12].

Dans le cadre de la chirurgie assistée par ordinateur, la méthode des éléments finis (§1.2.3) est utilisée pour représenter le comportement des tissus pendant l'intervention [26].

Les modèles déformables servent aussi d'un outil de modélisation géométrique [14, 92, 87]. Dans [87] on montre qu'à partir d'un modèle déformable d'une structure anatomique (vertèbre, ou visage) d'une personne quelconque, on peut reconstruire le modèle de cette structure chez n'importe quelle autre personne.

1.6 Conclusion

Un simulateur dynamique est un logiciel qui nécessite en général un temps d'exécution très important. Cette complexité vient soit du calcul des déformations [78, 43, 36] soit de celui de la collision [4, 72]. L'optimisation peut conduire dans certains cas à s'éloigner de la réalité physique en séparant le mouvement et les déformations par exemple [90, 35]. L'optimisation peut aussi se faire en ajoutant des restrictions particulières sur le comportement de l'objet: mouvement balistique, accélération limitée [80], objets parfaitement rigides, collisions peu fréquentes. C'est ce que l'on fait actuellement dans le domaine de la robotique [80, 4, 68].

Notre but initial était la prise en compte des interactions complexes entre le robot et son environnement (le cas d'un véhicule tout-terrain ou une main articulée). Dans ce cas nous sommes obligés de considérer des collisions très fréquentes (le cas du véhicule), des objets déformables (le cas des bouts des doigts de la main de Salisbury), des forces discontinues (contrôle et collision), des mouvements non balistiques (car le robot est contrôlé le long de sa trajectoire).

Pour cela nous avons choisi (§2) d'adapter, d'optimiser le système masses/ressorts (§1.2.5), et de l'intégrer avec la dynamique d'objets rigides (§1.2.1). Cela nous permet de calculer simultanément le mouvement et les déformations, de réduire la complexité du système (la complexité de système masses/ressorts est linéaire dans le cas général, et constante dans une architecture parallèle) et d'obtenir des com-

portements réalistes (le système masses/ressorts et la dynamique des objets rigides sont les deux développes dans le domaine de la mécanique [17, 30].

Pour détecter efficacement la collision nous avons adapté (§4.2) l'algorithme de Gilbert et al. (§1.3.1) car il permet de calculer la distance entre les enveloppes convexes de deux objets déformable.

La force de la collision est calculée (§4.3) en utilisant la méthode de pénalité non linéaire (§1.4.3) car elle fonctionne, à la fois, pour des objets rigides et déformable et car elle donne des résultats cohérents avec la réalité.

Dans les chapitres suivants, nous montrons comment nous avons adapté, optimisé le système masses/ressorts et comment l'avons intégré avec la dynamique d'objets rigides.

Chapter 2

Modèle et Modélisation

2.1 Introduction

Notre but est de définir un modèle (une représentation) permettant d'intégrer les trois comportements dynamiques principaux d'un objet (mouvement, déformations, et interactions). Ce modèle doit donc représenter la masse, la distribution de la masse, l'élasticité la viscosité, les facteurs du frottement et la forme géométrique de l'objet (car cette forme géométrique définit la zone d'interaction entre l'objet et son entourage). De plus, nous voulons que ce modèle soit adaptatif pour optimiser la complexité de la représentation en fonction de la nature de l'objet (rigide déformable), de sa forme géométrique (sphère, cube, etc...), et de la tâche à réaliser (une partie de l'objet ou l'objet tout entier va se déformer) pour s'approcher le plus possible de la complexité d'un modèle géométrique pur. Nous voulons aussi que le modèle soit simple pour que la tâche de la modélisation soit facile et automatisable.

Ce chapitre est constitué de deux parties principales. La première partie décrit le *modèle*. Elle montre la représentation dynamique de l'objet qui porte ses propriétés d'inertie et ses caractéristiques physiques (élasticité, viscosité et plasticité), sa représentation géométrique et la relation entre les deux représentations. La deuxième partie concerne la *modélisation* d'objets. Elle montre comment construire le modèle dynamique de l'objet à partir de sa forme géométrique et de ses propriétés physiques.

2.2 Description de notre modèle

Le modèle utilisé est une généralisation de la notion du système masses/ressorts [30], où un objet est représenté par un ensemble de masses ponctuelles (ou noeuds, figure 2.1) liées entre elles par des connecteurs viscoélastiques (type ressort/amortisseur) ou élasto-plastique, etc. Dans notre modèle, un noeud (ou une primitive) n'est pas forcement une masse ponctuelle, mais peut être n'importe quelle primitive dynamique dont le mouvement et les déformations peuvent être calculés par un modèle propre à elle. Par exemple une primitive peut être une particule, un objet rigide ou une chaîne articulée dont le mouvement est donné par le formalisme de Lagrange (§1.2.2). Pour le moment, **une primitive est un objet rigide** ou une masse ponctuelle comme cas particulier d'un objet rigide.

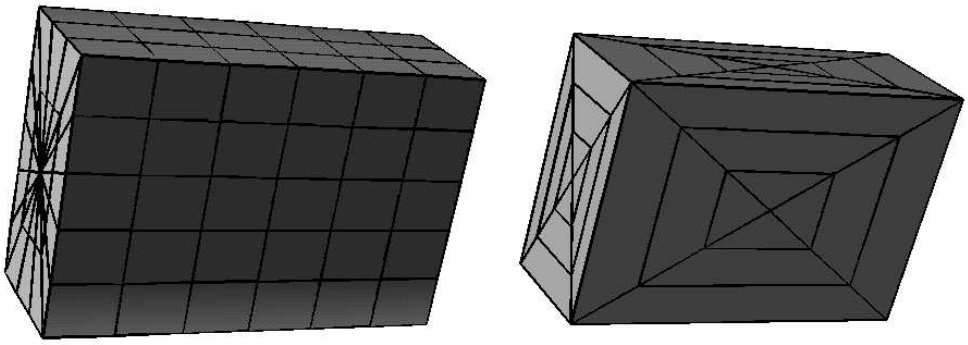


Figure 2.1: *Le système masse/ressort représente un objet par un ensemble de particules (les points d'intersection) liées entre elles par des ressorts.*

Cette généralisation a plusieurs avantages:

- elle permet d'optimiser le calcul du mouvement et des déformations de chaque primitive en utilisant le formalisme le plus efficace (on utilise la dynamique d'objets rigides pour simuler le comportement d'une primitive rigide et non pas le système masse/ressort);
- elle rend le modèle compatible avec d'autres modèles existants;
- elle permet une mise à jour facile du système en ajoutant d'autres types de primitives car le comportement dynamique de chaque primitive est calculé indépendamment.

2.2.1 Représentation géométrique

Pour que les différents objets puissent interagir entre eux (collision, frottement), il fallait qu'ils aient une représentation géométrique.

Plusieurs approches ont été proposées pour créer la forme géométrique d'un objet modélisé par le système masses/ressorts. Dans le système Cordis-Anima [72], on propose de lier la forme géométrique à chaque particule. La forme géométrique d'une particule est une sphère dont le centre est la particule même (figure 2.2). Cette représentation est très coûteuse car il faut augmenter le nombre de particules pour que la forme géométrique soit respectée même s'il ne s'agit pas d'un objet déformable. Pour optimiser la complexité de cette représentation, on propose [49] de faire une représentation par plusieurs couches. Une couche profonde ou "noyau", représente la partie peu déformable de l'objet. Une couche intermédiaire le "derme", représente la partie déformable de l'objet. Et une couche superficielle, l'"épiderme", exerce une pression sur les deux autres couches pour renforcer la cohésion interne (figure 2.2). Cette représentation a été aussi optimisée et automatisée par [20]. Cette optimisation consiste à trouver le nombre minimal de sphères capable de remplir la forme géométrique de l'objet et à ajouter automatiquement les connecteurs entre les particules.

Malgré ces optimisations, la complexité du modèle reste plus importante que celle des modèles géométriques classiques (basés sur des facettes) et le nombre de particules peut être important même pour des formes géométriques simples (figure 2.2).

D'autres approches consistent simplement à représenter la forme géométrique d'un objet discrétisé par des facettes triangulaires qui couvrent la surface extérieure de l'objet [45]. L'avantage des facettes triangulaires provient du fait qu'elles restent planes pendant les déformations de l'objet.

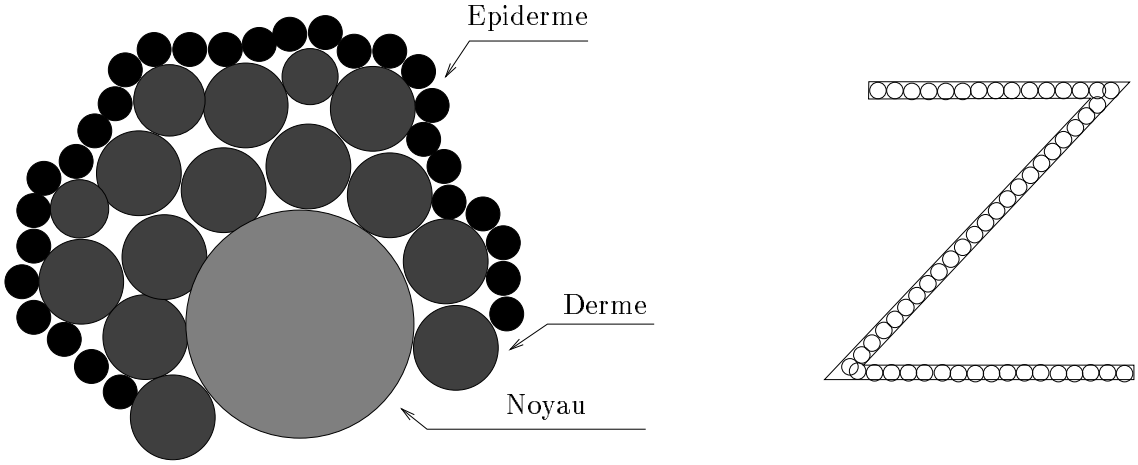
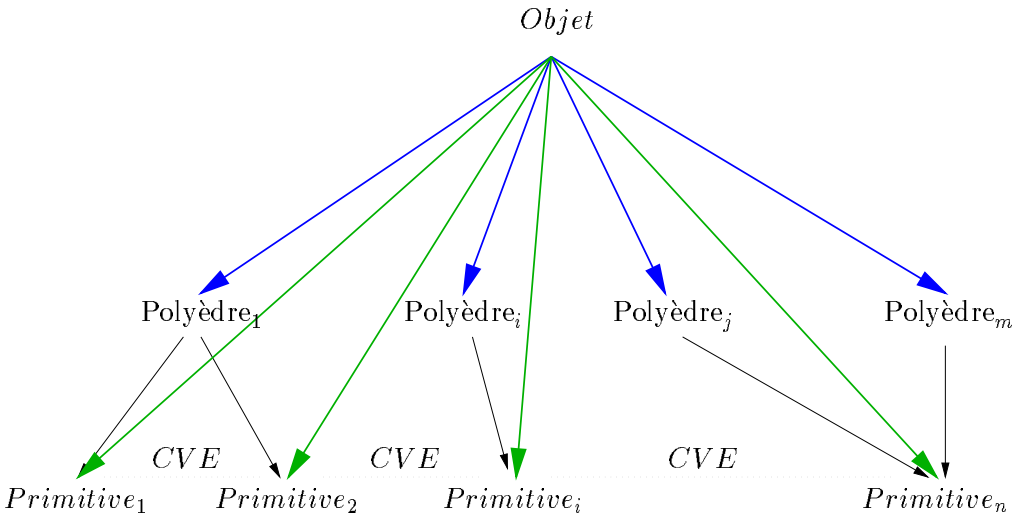


Figure 2.2: *Le modèle géométrique utilisé dans Cordis-Anima consiste à lier une zone sphérique à chaque particule. Ce modèle peut être optimisé en utilisant plusieurs couches à plusieurs définitions, mais le nombre de particules nécessaire pour s'approcher de la forme géométrique exacte reste relativement important.*

Notre but est d'avoir une représentation géométrique efficace, qui nous permet d'automatiser le passage au modèle dynamique. Pour cela, on a décidé que l'objet serait donné sous forme d'un ensemble de polyèdres convexes (figure 2.3). Ce choix a plusieurs avantages:

- la représentation est compatible avec d'autres modèles géométriques utilisés dans les domaines de la robotique et de la synthèse d'images. On peut passer d'une représentation polyédrique d'un objet à n'importe quel autre type de modélisation;
- il nous permet de bénéficier des nombreux algorithmes efficaces déjà développés pour manipuler des polyèdres. Par exemple plusieurs algorithmes efficaces ont été développés pour détecter le contact entre des polyèdres convexes (voir §4.2);
- La représentation polyédrique (la structure D.C.E.L [19]) est la représentation optimale (au niveau rapport mémoire/temps d'accès) qui nous permet d'avoir des informations complètes sur l'objet (sommets, arêtes, facettes, et leurs voisinages). Cela facilite la tâche d'automatisation et la mise à jour du système;
- il nous permet d'avoir un rendu réaliste.

Si une partie de l'objet est rigide, elle peut être représentée par une seule primitive rigide. Par contre sa forme géométrique se représente soit par un seul polyèdre convexe (s'il s'agit d'une partie convexe) soit par plusieurs polyèdres convexes (s'il s'agit d'une partie concave). Si la partie est déformable, elle sera représentée par un ensemble de primitives ponctuelles (particules). Sa forme géométrique se représente par un ou plusieurs polyèdres dont les sommets correspondent aux primitives.



CVE : Connecteurs viscoélastique

Figure 2.3: La forme géométrique d'un objet est représentée par un ensemble de polyèdres convexes, tandis que ses propriétés dynamiques sont représentées par un ensemble de primitives rigides. Les deux représentations ne sont pas indépendantes.

2.2.2 Représentation dynamique

La dynamique de l'objet dépend de sa masse, de la distribution de cette masse et de ses propriétés physiques (élasticité, viscosité, et plasticité).

Les déformations de l'objet changent la distribution de sa masse. Donc, la masse de l'objet ne peut pas être une propriété globale mais locale à chaque primitive. Pour que la représentation massique des différentes primitives soit cohérente, nous avons adopté une représentation qui marche pour les primitives les plus simples.

Chaque primitive est divisée en plusieurs particules (si ce n'est pas déjà le cas) et la masse de l'objet est divisée entre ces particules tout en respectant les propriétés d'inertie de l'objet (voir §6).

Cette discrétisation permet aussi de simplifier la construction du système, car on utilise les mêmes algorithmes pour calculer les matrices d'inertie, l'énergie mécanique et potentielle de n'importe quelle primitive. Ces algorithmes sont eux même très simples car ils sont appliqués sur un ensemble de particules, ce qui remplace les intégrales numériques complexes par des sommes finis.

La représentation dynamique de l'objet est alors une agglomération de particules. La distance relative entre les particules qui appartiennent à la même primitive est toujours constante car une primitive est par définition rigide. Par contre ce n'est pas le cas des particules qui appartiennent à deux primitives différentes. Il faut donc lier certaines d'entre elles afin d'assurer la connexité de l'objet tout en lui permettant de se déformer.

Les connecteurs lient deux particules de deux primitives différentes, mais non pas deux particules de la même primitive. Leurs comportements ressemblent à celui d'un ressort qui attache les différentes primitives.

2.2.3 Connexion

Dans le formalisme de base du système masse/ressort, il n'y a qu'un seul type de connecteurs dont le comportement ressemble à celui d'un ressort linéaire qui fixe la distance entre les deux particules. Ce connecteur est très pauvre dans le sens où la fixation de tous les degrés de liberté d'un objet nécessite un grand nombre de ces connecteurs. De plus, ce type de connecteur est mal adapté pour la modélisation des articulations prismatiques et rotoïdes. Il est cependant possible de les représenter en combinant plusieurs connecteurs linéaires (voir pour instance [72]), mais cela nécessite l'introduction de particules virtuelles qui peuvent affecter les propriétés de l'inertie de l'objet et augmenter inutilement la complexité du modèle. Pour cela on a développé et intégré dans notre modèle trois types de connecteurs:

Le **connecteur linéaire** (LS) qui connecte deux particules (a et b) afin de fixer la distance relative entre elles (figure 2.4.a). La force générée par ce connecteur est exprimée par l'équation suivante:

$$\vec{F_a} = (-\lambda \Delta p - \mu \dot{p})\vec{k_a}, \tag{2.1}$$

$$\dot{p} = (\dot{\vec{a}} - \dot{\vec{b}}).\vec{k_a}$$

où λ est la rigidité du connecteur, Δp est la variation de sa longueur (cette longueur dépend de la position d'équilibre p_0), μ le facteur d'amortissement, $\dot{p}$ la vitesse relative de la particule a par rapport à la particule b, et $\vec{k_a} = \frac{\vec{b}-\vec{a}}{|\vec{b}-\vec{a}|}$. Ce connecteur contraint un degré de liberté seulement par particule.

Le **connecteur angulaire** (TS) est développé afin d'associer des contraintes angulaires à un ensemble de trois particules (a, b et c). La force appliquée par ce connecteur sur une particule non-centrale est exprimée par l'équation suivante:

$$\vec{F_b} = (-\lambda \Delta\theta - \mu \dot{\theta})\vec{k_b}$$

$$\dot{\theta} = \frac{1}{sin\theta |\vec{ca}||\vec{cb}|}\left(\frac{|\vec{ca}|}{|\vec{cb}|} cos\theta(\vec{cb}.\dot{\vec{cb}}) + \frac{|\vec{cb}|}{|\vec{ca}|} cos\theta(\vec{ca}.\dot{\vec{ca}}) - \vec{ca}.\dot{\vec{cb}} - \vec{cb}.\dot{\vec{ca}}\right)$$

où $\Delta\theta$ est la variation de l'angle formée par les trois particules, (elle dépend de la valeur angulaire à l'équilibre θ_0), $\dot{\theta}$ la vitesse angulaire, $\vec{ca} = \vec{a} - \vec{b}$, $\dot{\vec{ca}}$ est la dérivée (ou la vitesse) du vecteur $\vec{ca}$, et $\vec{k_b}$ est un vecteur unitaire perpendiculaire à la direction définie par les deux particules en question, la figure 2.4.b l'illustre. A

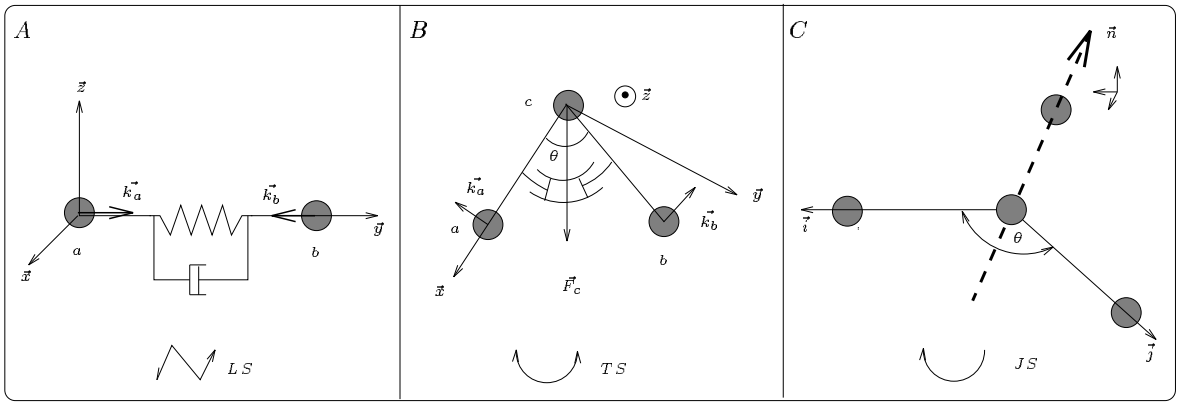


Figure 2.4: *Les connecteurs LS, TS et JS.*

cause du principe de l'action/réaction, la force $\vec{F_c}$ appliquée sur la particule centrale est égal à l'opposé de la somme de $\vec{F_b}$ et $\vec{F_a}$.

Ce connecteur contraint trois degrés de liberté (un degré par particule). Mais quand l'angle θ_0 de ce connecteur est égal à 180°, ce connecteur contraint deux degrés de liberté par particule. Cette propriété est très utile quand on représente une barre ou une articulation prismatique (fig. 5.13). Dans le cas général ce connecteur sert à représenter les changements morphologiques d'un objet. L'inconvénient d'un tel connecteur est la symétrie des deux particules non-centrales par rapport à la troisième particule. Cette symétrie nous empêche de distinguer l'angle θ de l'angle $\theta + \pi$, ce qui peut causer des problèmes quand on contrôle une articulation rotoïde. En fait les connecteurs servent à engendrer des torques dont le rôle est d'amener l'articulation à une position voulue. Si la distance entre la position actuelle de l'articulation et la position finale est supérieure à 180^o, ce connecteur ne permet pas de formuler une telle requête.

La représentation des contraintes angulaires par trois ressorts linéaires qui lient les trois particules n'est pas correcte, car d'une part il nous oblige à fixer la distance entre les particules (ce qui n'est pas forcément le but), d'autre part la distance entre les deux particules non-centrales change en fonction de l'angle.

Le **connecteur articulaire** (JS) est développé pour éviter les inconvénients de la symétrie du connecteur TS et pour être capable de représenter des articulations rotoïdes (fig. 5.13). Le connecteur JS (fig. 2.4.a) est un connecteur TS muni d'une quatrième particule qui permet de définir l'axe et la direction positive de la rotation. La force appliquée par ce connecteur sur une particule non-centrale est exprimée par l'expression suivante:

$$\vec{F_a} = (-\lambda(\theta - \theta_0) - \mu\dot{\theta})\vec{k_a} \tag{2.2}$$

$$\dot{\theta} = \frac{1}{sin\theta|\vec{ca}||\vec{cb}|}(\frac{|\vec{ca}|}{|\vec{cb}|}cos\theta(\vec{cb}.\dot{\vec{cb}}) + \frac{|\vec{cb}|}{|\vec{ca}|}cos\theta(\vec{ca}.\dot{\vec{ca}}) - \vec{ca}.\dot{\vec{cb}} - \vec{cb}.\dot{\vec{ca}})$$

où,

$$\theta = \begin{cases} acos(2\pi - \vec{i} \times \vec{j}) & \text{si } (\vec{i} \wedge \vec{j}) \times \vec{n} < 0 \\ acos(\vec{i} \times \vec{j}) & \text{sinon} \end{cases} \quad \vec{k_a} = \vec{n} \wedge \vec{j}$$

pour respecter le principe de l'action/réaction, la force appliquée sur la particule centrale est égale à l'inverse de la somme des forces appliquées sur les deux autres particules.

Les articulations rotoïdes sont des combinaisons des connecteurs linéaires et articulaires (figure 2.5). Pour changer dynamiquement la valeur actuelle de cette articulation, il suffit de changer la valeur θ_0 du connecteur articulaire. Cela engendre, d'après l'équation 2.2, des forces qui ramènent l'articulation à la position voulue. Les articulations prismatiques sont des combinaisons des connecteurs linéaires et angulaires (figure 2.5). Pour changer dynamiquement la valeur actuelle de cette articulation, il suffit de changer la valeur P_0 du connecteur linéaire. Cela engendre, d'après l'équation 2.1, des forces qui ramènent l'articulation à la position voulue. Le rôle des connecteurs angulaires est d'assurer la colinéarité des différentes parties.

Alors, Le modèle dynamique des articulations rotoïdes et prismatique n'est pas passif mais il sert à contrôler effectivement le mouvement de ces articulations.

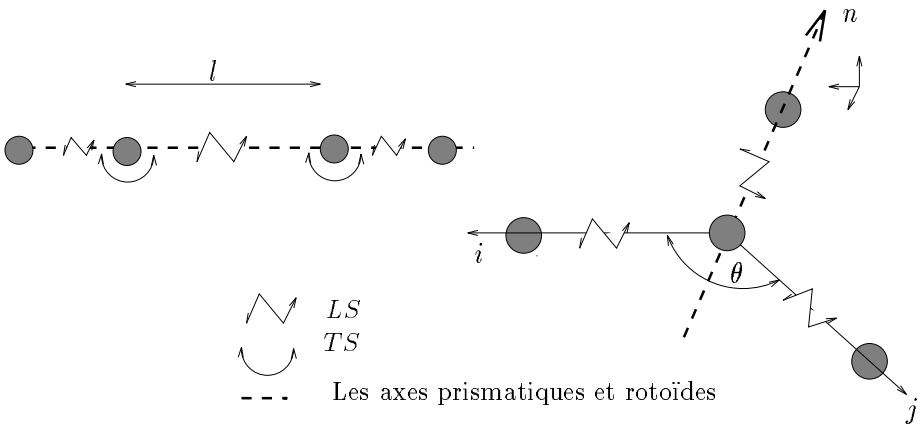


Figure 2.5: *Les articulations prismatiques et rotoïdes.*

2.2.4 Effet des paramètres physique λ et μ sur le comportement d'un objet

Le comportement dynamique des particules liées par un connecteur dépend des valeurs de λ et μ. La figure 2.6 montre l'evolution de la position relative de deux particules liées par un ressort, en fonction des paramètres λ et μ de ce ressort. Il faut remarquer que cet effet dépend aussi des masses des particules et des masses d'autres particules dont elles sont liées. On n'a donc pas une méthode numérique capable de nous trouver les valeurs correspondant le mieux à nos besoins. Pour trouver ces valeurs il faut faire appel à des approches d'optimisations (§6.4).

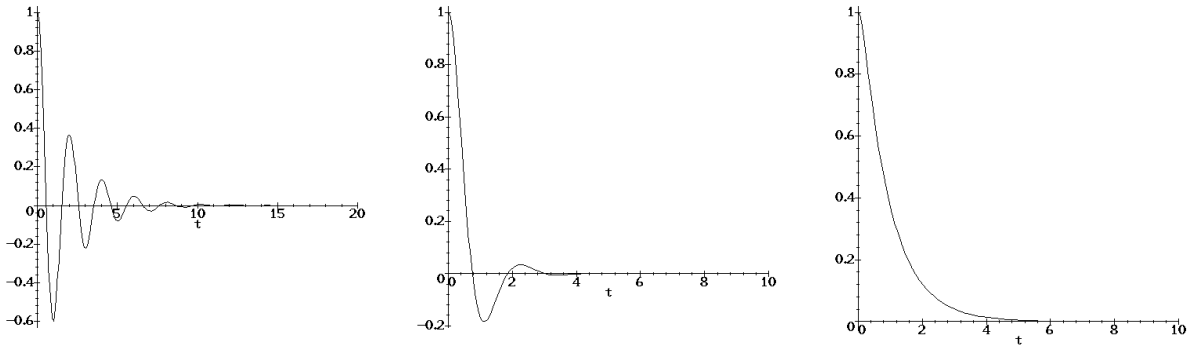


Figure 2.6: *L'evolution de la position relative de deux particules en fonction des paramètres physiques (λ et μ) du ressort qui les connecte. De gauche à droite: $\lambda >> \mu$ dans ce cas il n'y a pas d'amortissement suffisant et le système oscille beaucoup avant de se stabiliser; $\lambda \simeq \mu$: dans ce cas l'amortissement est suffisant pour que le système atteigne rapidement la stabilisation; $\lambda << \mu$: dans ce cas l'amortissement est suffisamment important pour que le système atteigne la stabilisation sans dépassement.*

Exemple: Un drap est représenté en utilisant à la foi des connecteurs linéaires et des connecteurs angulaires. La figure 2.7 donne une représentation schématique d'un tel objet avec un petit nombre de particules. Les paramètres des connecteurs angulaires sont λ_a et μ_a et les paramètres des connecteurs linéaires sont λ_l et μ_l.

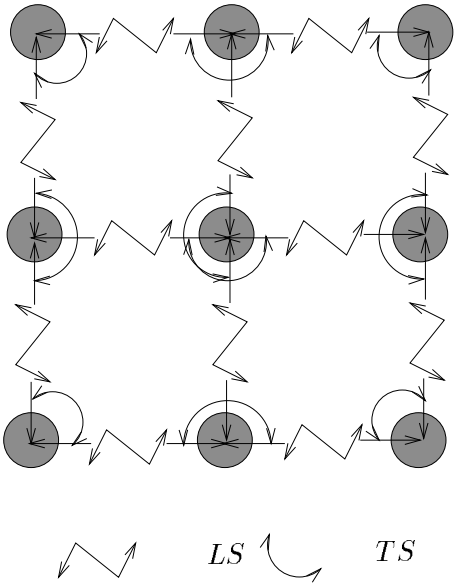


Figure 2.7: *Une représentation possible d'un drap.*

Si on applique une force dirigée vers le haut, sur le milieu de ce drap. Le drap se déforme de manière isotropique (figure 2.8). Si les connecteurs angulaires sur un axe sont plus rigides que les autres, cela donne un comportement non isotropique, bien que la même force soit appliquée (figure 2.9). En éliminant les connecteurs angulaires, le drap devient parfaitement flexible, mais il se comporte comme un ensemble de carrés rigides articulés et on constate l'apparition d'arêtes vives (figure 2.10). Pour éviter les arêtes vives, on rajoute des connecteurs angulaires avec les rigidités λ_a nulles et des viscosités μ_a assez importantes. Cela rend les vitesses relatives des deux particules voisines très proches et donne au drap un comportement lisse tout en restant parfaitement flexible (figure 2.11). En éliminant les connecteurs linéaires, le drap devient parfaitement déformable, sa surface reste toujours lisse à cause des connecteurs linéaires, mais les vitesses relatives entre les différentes particules sont très variées (figure 2.12). Pour minimiser les vitesses relatives entre les particules voisines, et pour obtenir un comportement plus réaliste, on rajoute des connecteurs linéaires dont les rigidités λ_l sont nulles, et dont les viscosités μ_l sont assez importantes (figure 2.13).

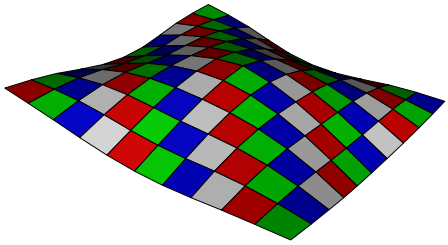


Figure 2.8: $\lambda_a \neq 0$, $\lambda_l \neq 0$, $\mu_a \neq 0$ *et* $\mu_l \neq 0$

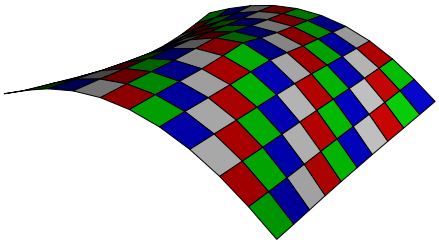


Figure 2.9: λ_a *a différentes valeurs selon l'axe.*

2.3 La modélisation d'objets

Etant donné un objet quelconque, la création d'un modèle dynamique de cet objet passe par deux étapes. La première consiste à créer une représentation dynamique paramétrée de l'objet qui respecte sa forme géométrique. La deuxième consiste à calibrer cette représentation afin de respecter les propriétés dynamiques de cet objet (centre d'inertie, matrice d'inertie, élasticité, plasticité, viscosité, etc...).

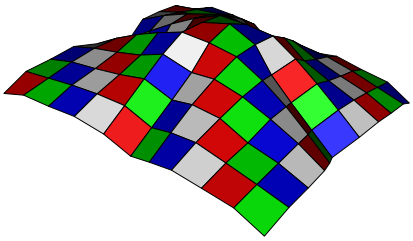


Figure 2.10: $\lambda_a = 0$, $\lambda_l \neq 0$, $\mu_a = 0$ et $\mu_l \neq 0$.

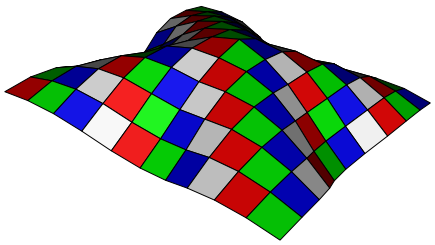


Figure 2.11: $\lambda_a = 0$, $\lambda_l \neq 0$, $\mu_a \neq 0$ et $\mu_l \neq 0$.

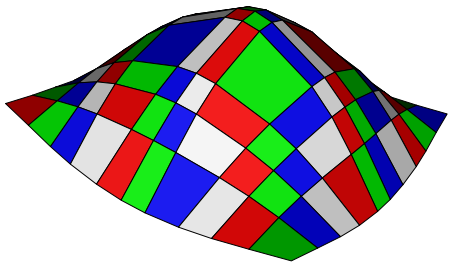


Figure 2.12: $\lambda_a \neq 0$, $\lambda_l = 0$, $\mu_a \neq 0$ et $\mu_l = 0$

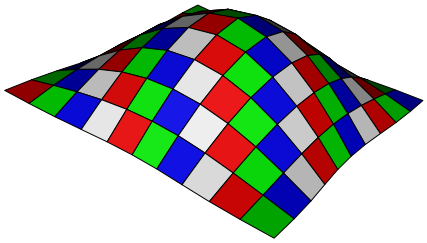


Figure 2.13: $\lambda_a \neq 0$, $\lambda_l = 0$, $\mu_a \neq 0$ et $\mu_l \neq 0$

Dans la suite, nous expliquons comment automatiser la première étape, tandis que la deuxième étape va être expliquée plus tard dans le chapitre concernant l'identification § 6.

2.3.1 Problématique

Le passage du modèle géométrique d'un objet et de ses propriétés physiques (masse, élasticité, viscosité et plasticité) à son modèle dynamique ne peut pas être fait, dans le cas général, d'une manière purement automatique pour plusieurs raisons:

- le modèle géométrique ne contient pas toutes les informations nécessaires pour la création d'un modèle dynamique, telles que la masse, la distribution de cette masse à l'intérieur de l'objet, son élasticité, sa plasticité, sa viscosité, etc...;

- même avec ces informations une méthode automatique générale ne peut pas être optimale car elle ne peut pas prendre en compte la fonction de l'objet. Si l'objet subit pendant son utilisation des efforts très importants par rapport à sa rigidité, alors il va se déformer. Cela nécessite une représentation plus précise pour obtenir un comportement correct, sinon on peut simplifier la représentation de cet objet pour gagner au niveau temps d'exécution; De plus on peut parfois affiner la représentation de certaines parties de l'objet, qui sont plus critiques sur le plan fonctionnel.

La création manuelle d'un modèle dynamique pose aussi certains problèmes car elle demande une bonne expertise pour choisir les positions et les masses des différentes particules. Pour cela on propose une méthode semi-automatique pour créer une représentation dynamique d'un objet.

2.3.2 Du modèle géométrique au modèle dynamique

Etant donné le modèle géométrique d'un objet sous forme d'un ensemble de polyèdres convexes, la création du modèle dynamique consiste à diviser chaque polyèdre (figure 2.14) en un ensemble de particules [54] qui doivent respecter certains critères pour que la représentation de l'objet soit efficace et que le calibrage de ce modèle soit possible:

- la masse doit être distribuée entre les différentes particules, de façon à ce que la matrice d'inertie et le centre d'inertie du polyèdre soient respectés.

- les positions des particules doivent être choisies de sorte que leur enveloppe coïncide avec la forme géométrique de l'objet. Cela implique au minimum de placer une particule sur chaque sommet de l'objet.

- les particules doivent être distribuées selon les axes principaux de l'objet. Normalement les déformations sont plus importantes quand on agit sur un axe principal d'un objet, la distribution des particules sur les axes principaux permet de mieux contrôler leurs déformations et par conséquent de minimiser l'erreur commise.

- le nombre de ces particules dépend seulement de la déformabilité du polyèdre et de sa forme géométrique (il ne dépend pas de ses propriétés d'inertie). Les objets les plus déformables sont représentés par un nombre plus grand de particules.

- le positionnement de ces particules dans l'espace permet de définir des relations de voisinage entre elles pour pouvoir ajouter automatiquement des connecteurs entre les particules voisines et ainsi fixer tous les degrés de liberté de l'objet.

Chaque polyèdre doit être homogène (c'est-à-dire que la masse est uniformément distribuée, et que le polyèdre est soit entièrement rigide soit entièrement déformable). Dans le cas où le polyèdre est rigide, les particules forment une seule primitive et il n'y a pas de connecteur à ajouter. Dans le cas contraire il faut automatiquement ajouter des connecteurs entre elles.

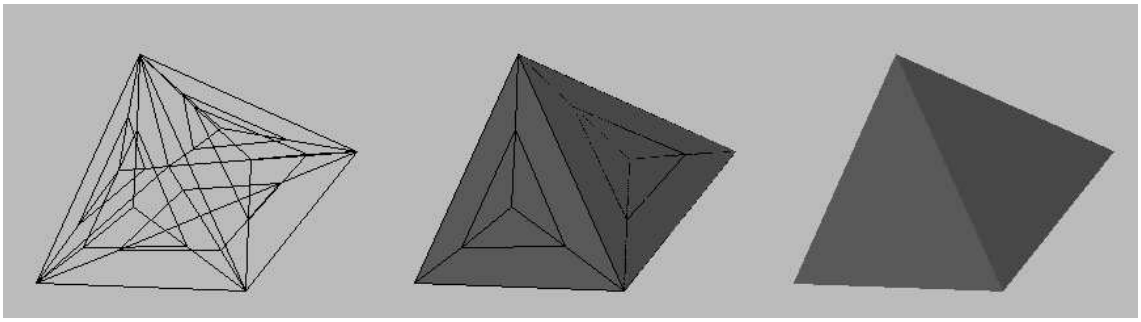


Figure 2.14: *Ce polyèdre est représenté par un nombre fini de particules qui respectent ses propriétés d'inertie. Les sommets de ce polyèdres sont attachés avec les particules qui le constituent, donc, le polyèdre se déforme quand ces particules bougent.*

2.3.3 La discrétisation automatique et adaptative d'un polyèdre

Il existe plusieurs méthodes pour mailler un polyèdre [37]. Une discrétisation uniforme ne satisfait pas les conditions mentionnées ci-dessus, car il nous faut un grand nombre de particules pour respecter les propriétés de l'inertie d'un polyèdre ainsi que sa forme géométrique (figures 2.15).

La méthode de Delaunay-Voronoi [13] consiste à partir des points de contour de départ, d'ajouter d'autre points sur la surface du maillage, puis d'effectuer une première tetraèdrisation. On rajoute des points supplémentaires à l'intérieur du maillage de sorte que ces points aillent en priorité aux endroits où le maillage est peu dense (figure 2.16).

Les méthodes de superposition semblent être les plus régulières et les plus symétriques par rapport au centre de gravité d'un objet. Le maillage par transport-projection consiste, à partir d'un maillage d'un objet simple (triangle, tétraèdre, hexaèdre), à trouver un homomorphisme entre cet objet simple et l'objet à mailler. Cet homomorphisme lie chaque point de l'objet régulier avec son correspondant de l'objet à mailler et permet, par conséquent, de trouver le maillage correspondant (figure 2.17).

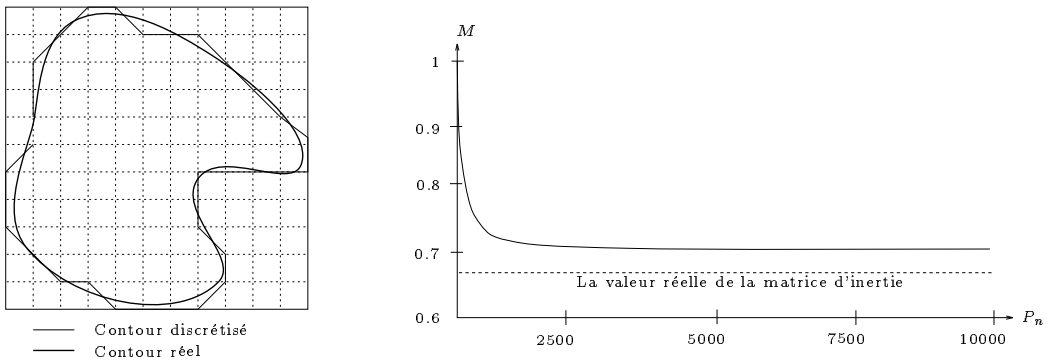


Figure 2.15: *La discrétisation uniforme, nécessite un grand nombre de particules pour obtenir une bonne approximation de la forme géométrique de l'objet d'origine. Même quand la forme géométrique est simple, il faut un grand nombre de particules pour respecter les propriétés d'inertie. Cette figure montre la variation d'un paramètre de la matrice d'inertie en fonction du nombre de particules utilisées P_n.*

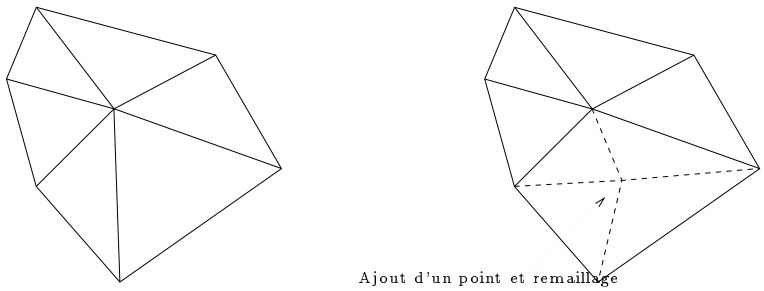


Figure 2.16: *Le maillage d'une surface 2D par la méthode de "Delaunay-Voronoi"*

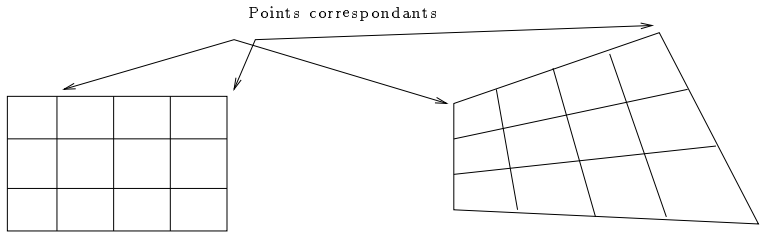


Figure 2.17: *Le maillage par superposition-projection*

Les méthodes frontales permettent de construire le recouvrement de l'objet à partir de son contour discrétisé (figure 2.18).

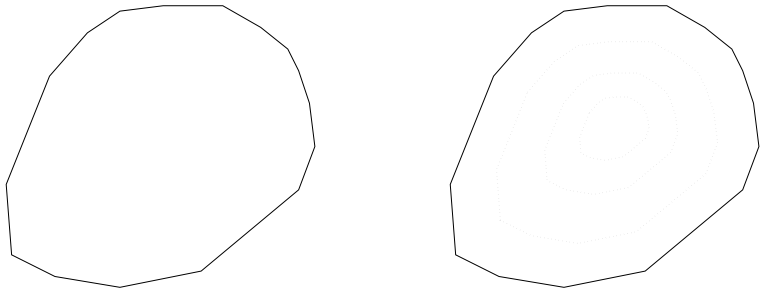


Figure 2.18: *Le maillage par propagation d'ondes*

Méthode retenue

Notre méthode de maillage est une combinaison d'une méthode frontale et d'une méthode de superposition-projection. On commence par discrétiser les facettes du polyèdre puis on discrétise son volume. On distingue plusieurs cas:

Facette polygonale: On commence par discrétiser le contour de cette facette en mettant d'abord une particule sur chaque sommet puis en discrétisant régulièrement chaque arête. La discrétisation d'une facette passe par la discrétisation d'un polygone régulier (toutes les arêtes et tous les angle sont égaux) qui a le même nombre d'arêtes. Si la facette a quatre arêtes, le polygone régulier correspondant est un carré que l'on discrétise d'une façon matricielle et on projette la discrétisation sur la facette (figure 2.19). Dans le cas contraire, on discrétise le polygone régulier en utilisant une méthode frontale. Dans ce cas, les ondes partent du contour de la facette et se trouvent à son centre géométrique (figure 2.19). Le maillage obtenu va donc être projeté sur la facette de départ.

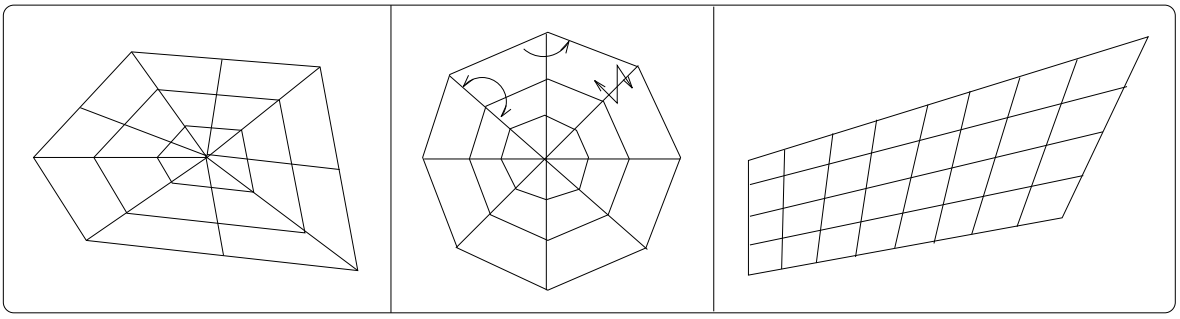


Figure 2.19: *La discrétisation d'une facette polygonale convexe*

Hexaèdre: Un hexaèdre est un polyèdre qui a six facettes. Pour discrétiser un hexaèdre, on commence par discrétiser d'une manière matricielle un cube puis on projette la discrétisation sur l'hexaèdre.

Un polyèdre de balayage: c'est un polyèdre défini par deux facettes. On commence par discrétiser chaque facette, puis on extrapole les lignes liant chaque point d'une facette avec le point correspondant de l'autre. Cette méthode de discrétisation est très adaptée pour représenter des câbles (figure 2.20).

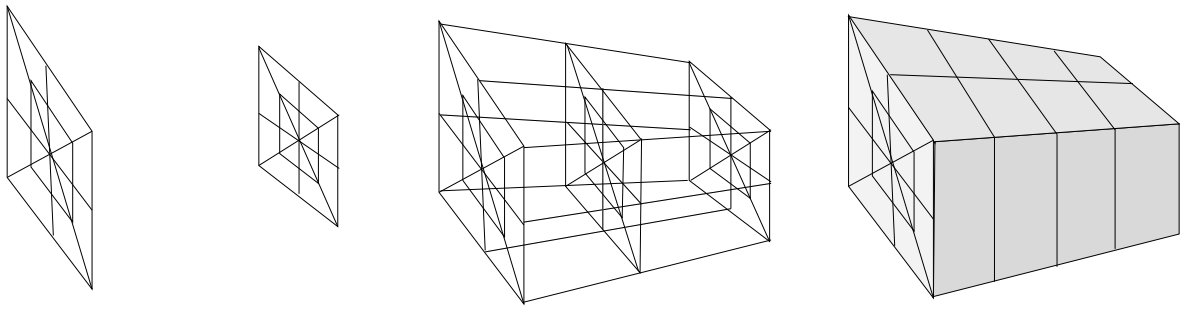


Figure 2.20: *La discrétisation d'un hexaèdre*

Polyèdre quelconque: pour discrétiser un polyèdre quelconque, on commence par discrétiser chacune de ses facettes, puis on ajoute des couches internes en utilisant le principe de propagation d'ondes où les ondes convergent vers le centre d'inertie de l'objet. (figure 2.21).

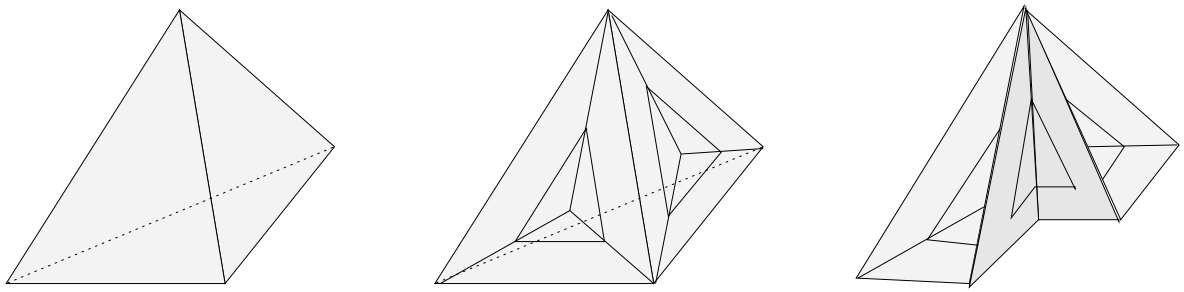


Figure 2.21: *La discrétisation d'un polyèdre non hexagonal.*

Dans le cas d'un polyèdre rigide il suffit de le représenter par une particule à chaque sommet, une particule au centre d'inertie de chaque facette (pour que les facettes soient triangulaires), et une particule à son centre d'inertie. La distribution de la masse entre les différentes particules, afin de respecter les propriétés d'inertie de l'objet, sera expliquée dans le chapitre de l'identification (§ 6.2).

La méthode proposée est adaptative, car on peut contrôler le nombre des particules en fonction de la déformabilité du polyèdre, ainsi que leurs positions en fonction de la tâche. Par exemple le milieu d'une table est plus sensible aux efforts extérieurs que son côté, et par conséquent il se déforme souvent plus facilement. Il vaut mieux, pour cette raison, mettre plus de particules au milieu de la table que sur les cotés

(figure 2.22). Le choix des positions des particules se fait pendant l'extrapolation. Pour extrapoler n points P_i entre deux points de référence P_1 et P_2 on utilise la formule suivante:

$$P_i = P_1 + \frac{i}{n}^k (P_2 - P_1) | i \in [0..n]$$

$k = 1$: la distribution de p_i est uniforme entre P_1 et P_2 $k < 1$: les points sont plus proche de P_2 $k > 1$: les points sont plus proche de P_1.

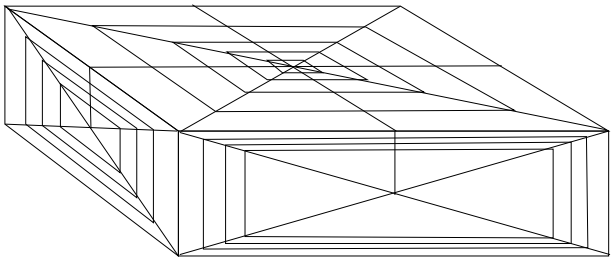


Figure 2.22: *Une discrétisation adaptative du plateau d'une table: La facette supérieure du plateau est discrétisée en un grand nombre de particules car c'est elle qui subit les déformations les plus importantes. Les particules sont concentrées vers le milieu du plateau, car le milieu se déforme normalement plus que les côtés.*

2.3.4 Insertion des connecteurs dans le modèle discrétisé

Une fois que les polyèdres sont discrétisés en plusieurs particules, il faut ajouter des connecteurs entre les particules afin d'obtenir un objet connexe. La connexion de deux particules qui appartiennent à deux différents polyèdres se fait manuellement. Cette liaison peut se faire de trois manières différentes :

- si deux polyèdres sont adjacents, on peut fusionner les particules qui sont en contact. Cette fusion consiste à remplacer les deux particules par une troisième dont la masse est la somme de leurs masses, et qui est connectée avec toutes les particules qui étaient connectées avec elles. Cela ne peut être fait que si l'un de ces polyèdres est déformable car on ne peut pas fixer la position d'une particule par rapport à deux repères mobiles et indépendants;

- si les deux particules appartiennent à des polyèdres rigides, ou si les deux particules n'ont pas la même position, dans ce cas la liaison se fait en ajoutant un connecteur entre elles;

- si deux polyèdres rigides sont liés par plus de deux particules non colinéaires, on peut les unifier par une seule primitive rigide.

Si le polyèdre est déformable, chaque particule de ce polyèdre forme une primitive. La connexion entre ces primitives peut se faire d'une manière automatique. Etant donné un polyèdre discrétisé de la manière expliquée ci-dessus, et trois connecteurs (linéaire, angulaire, et articulaire), la connexion se fait de la manière suivante:

- il faut connecter chaque paire de particules voisines par un connecteur linéaire (figure 2.19). Deux particules sont voisines s'il y a une arête du maillage qui les lie. Ces connecteurs sont suffisants pour fixer tous les degrés de liberté de l'objet. Mais ces connecteurs agissent dans un seul sens et ils n'ont aucun effet dans les autres sens ce qui fait que la surface de l'objet peut avoir un comportement chaotique et discontinue (figure 2.23). C'est pourquoi on ajoute d'autres types de connecteurs;

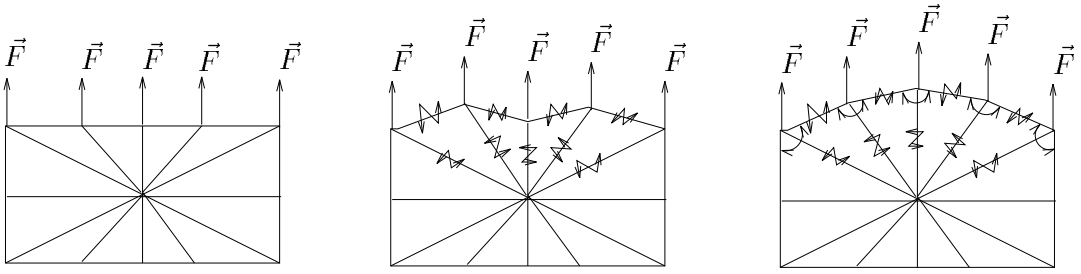


Figure 2.23: *Les connecteurs linéaires ont des réactions différentes vis à vis des forces extérieures. Pour obtenir un comportement plus réaliste, il faut ajouter des connecteurs angulaires et articulaires.*

- il faut connecter chaque triple particules colinéaires et voisines par un connecteur angulaire. Ces connecteurs se comportent comme un filtre, ils rendent les déformations de l'objet plus continues et ils nous permettent d'obtenir des déformations plus réalistes;

- il faut connecter chaque triple particules voisines et non colinéaires par un connecteur articulaire (figure 2.19). Le but est d'avoir des déformations continues, mais l'avantage de mettre des connecteurs articulaires, est qu'ils régissent correctement même si les déformations sont importantes.

Un objet réel contient un très grand nombre de particules. Donc il faut imaginer que dans notre modèle la distance entre deux particules, qui est relativement grande, est remplie par d'autres particules et que le ressort qui lie ces deux particules est en vérité l'équivalent de plusieurs ressorts liés en parallèle. Il faut donc prendre en compte la distance entre les particules pour déterminer le coefficient du ressort qui les lie. Soient λ_0 et μ_0 les coefficients d'un ressort dont la longueur est égale à l'unité de mesure, les coefficients d'un ressort de longueur quelconque l sont (§**??**): $\lambda = \frac{\lambda_0}{l}$ et $\mu = \frac{\mu_0}{l}$. Cela nous permet de récompenser l'effet de la discrétisation sur la fidélité du modèle afin d'obtenir une présentation qui est indépendante, le plus possible, du niveau de la discrétisation.

2.4 Conclusion

Notre modèle dynamique est une généralisation de la notion du système masse/ressort, dans lequel un objet est représenté par un ensemble de masses ponctuelles liées entre

elles par des connecteurs viscoélastiques de type ressort/amortisseur. Cette généralisation consiste d'une part, à remplacer les particules par des primitives mécaniques plus complexes (les objets rigides, et les particules restent un cas particulier d'un objet rigide) et d'autre part, à ajouter des nouveaux types de connecteurs (les connecteurs angulaires et articulaires) pour mieux contrôler le comportement de l'objet. Cela nous a permis d'atteindre le but voulu et d'optimiser le temps d'exécution en traitant les parties rigides d'un objet comme une seule unité, et de bénéficier des études déjà faites sur la dynamique d'objets rigides concernant sa relation avec le monde réel.

La représentation géométrique d'un objet est choisie pour être compatible avec d'autres modèles existants, et pour pouvoir bénéficier des algorithmes déjà développés concernants surtout la détection du contact. Donc, un objet est représenté par un ensemble de polyèdres initialement convexes. Un polyèdre peut correspondre à une primitive (s'il s'agit d'une partie rigide et convexe de l'objet), à plus d'une primitive (s'il s'agit d'une partie déformable de l'objet), ou à une partie d'une primitive (s'il s'agit d'une partie concave et rigide de l'objet).

La représentation dynamique est choisie pour qu'elle soit homogène quel que soit le type de la primitive et pour optimiser les opérations dynamiques (calcul de la matrice d'inertie et du centre d'inertie). Donc chaque primitive est dynamiquement représentée par plusieurs particules et sa masse est distribuée entre elles afin de respecter les propriétés de l'inertie de cette primitive.

Pour assurer la connexité de l'objet, les différentes primitives doivent être liées entre elles tout en permettant à l'objet de se déformer. La connexion des ces particules se fait à l'aide des connecteurs dont le comportement ressemble à celui d'un ressort. Trois types de connecteurs ont été développés pour mieux contrôler le comportement de l'objet et pour pouvoir représenter des articulations rotoïdes et prismatiques.

Le passage entre le modèle géométrique d'un polyèdre à son modèle dynamique peut se faire d'une manière semi-automatique et adaptative. Elle est semi-automatique car la discrétisation en particules et la distribution de la masse entre elles (afin de respecter les propriétés de l'inertie) peuvent se faire automatiquement tandis que le nombre de ces particules doit être choisi par l'utilisateur. Elle est adaptative car le nombre de particules dépend seulement de la déformabilité et de la forme géométrique de ce polyèdre. Le nombre minimal de particules qui permet de modéliser un objet rigide dépend de sa forme géométrique (une particule par sommet). Donc, la complexité du modèle dynamique s'approche de celle du modèle géométrique quand l'objet est rigide.

Chapter 3

Mouvement et Déformations

3.1 Introduction

Le mouvement et les déformations d'un objet sont les résultats des mouvements de ses primitives qui sont des objets rigides discrétisés (une particule est un cas particulier d'un objet rigide). Le mouvement d'une primitive est caractérisé par les lois de la dynamique et donné par la résolution d'une équation différentielle qui lie la position et la vitesse d'un objet avec les forces extérieures agissantes sur elle. Les différentes équations différentielles qui caractérise le mouvement des différentes primitives ne sont pas indépendantes à cause des forces d'interaction entre elle (il y a les forces engendrées par les connecteurs viscoélastiques qui lient deux primitives du même objet et les forces de collision et frottement engendrées par l'interaction entre deux primitives de deux objets différents). De plus, ces équations sont discontinues à cause de la force de la collision et la force du frottement. Sous ces conditions, la résolution de l'équation du mouvement devient difficile et nécessite parfois une discrétisation temporaire très fine ce qui peut ralentir énormément l'exécution.

Pour optimiser le temps d'exécution, on a développé une approche adaptative basée sur la notion d'énergie mécanique. Cette approche permet également d'estimer l'erreur commise et d'éviter la divergence numérique.

Ce chapitre explique comment trouver les équations du mouvement, comment les résoudre et comment optimiser le temps de calcul par l'approche adaptative proposée.

3.2 Equation du mouvement

Le mouvement d'une primitive peut être vue comme la combinaison de deux mouvements: translation puis rotation autour du centre d'inertie. Le mouvement de translation est caractérisé par la loi de Newton:

$$\vec{F} = m\vec{\gamma}$$

où,
$\vec{F}$ est la somme des forces externes, m la masse de la primitive, et γ son accélération qui est égale à la dérivée seconde de la position du centre d'inertie $\vec{\gamma} = \ddot{\vec{P}}$.

Le mouvement de rotation est caractérisé par l'équation d'Euler:

$$\vec{N} = I\dot{\vec{W}} + \vec{W} \wedge (I.\vec{W})$$

où,
$\vec{N}$ est la somme des moments externes, I est la matrice d'inertie de l'objet, et $\vec{W}$ est la vitesse de la rotation autour du centre d'inertie de la primitive. Si l'orientation de la primitive est donnée par le vecteur $\Theta = (\theta_x, \theta_y, \theta_z)$, alors $|\vec{W}| = |\dot{\Theta}|$. Quand la primitive est une particule, $\vec{W}$ est toujours nulle. Comme les primitives sont liées par des connecteurs, les équations différentielles précédantes sont liées, et elles donnent le système suivant:

$$\begin{aligned}
&\vec{F_i} = m_i\vec{\gamma_i} \\
&\vec{N_i} = I_i\dot{\vec{W}_i} + \vec{W_i} \wedge (I_i.\vec{W_i}) \\
&\vec{F_i} = f(\vec{P_j}, \dot{\vec{P}_j}, \vec{\Theta_j}, \vec{W_j}, \vec{FI_i}, \vec{FC_i}) | i, j \in [1..n] \\
&\vec{N_i} = g(\vec{F_i})
\end{aligned} \tag{3.1}$$

où n est le nombre de primitives, $\vec{P_i}$ la position du centre d'inertie de la primitive i à l'instant t, m_i sa masse, $\dot{\vec{P}_i}$ la vitesse de son centre d'inertie, Θ_i l'orientation de la primitive, $\dot{\vec{W}_i}$ sa vitesse de rotation autour de son centre d'inertie, $\vec{F_i}$ est la somme des forces appliquées sur la primitive i (cette force est une fonction f de $\vec{P_j}, \dot{\vec{P}_j}, \vec{\Theta_j}$ *et* $\vec{W_j}$, où j est une primitive parmi les n primitives dans la scène; elle dépend aussi de la force d'interaction $\vec{FI}$ et la force de contrôle $\vec{FC}$ appliquées sur la primitive i), et $\vec{N_i}$ est la somme des moments externes appliqués sur la primitive i, ces moments sont une fonction g des forces externes. Le mouvement et les déformations de l'objet peuvent être trouvés en résolvant ce système.

3.3 Résolution de l'équation du mouvement

Pour résoudre une équation différentielle, il y a deux méthodes de base [2]:

Différence finie: Etant donnée une fonction U_t, la différence finie exprime les dérivées de U_t en fonction de U_t. Par exemple, Houbolt donne le chemin implicite[1] suivant:

$$\begin{aligned}
\dot{U} &= \tfrac{1}{6\tau}(11U_{t+\tau} - 18U_t + 9U_{t-\tau} - 2U_{t-2\tau}) \\
\ddot{U} &= \tfrac{1}{\Delta t^2}(2U_{t+\tau} - 5U_t + 4U_{t-\tau} - U_{t-2\tau})
\end{aligned} \tag{3.2}$$

Cette solution est stable quelque soit la valeur de τ.

Développement limité: Cette méthode exprime la valeur d'une fonction U_t en fonction de ses dérivées:

$$U_{t+\tau} = \sum_0^n \frac{\tau^n}{n!} U_t^{(n)} + O(\frac{\tau^{n+1}}{(n+1)!} U_t^{(n+1)})$$

[1] Si $P_{t+\tau} = f(P_x) \mid x <= t$ l'approche est explicite (incrémentale), sinon il est implicite.

Si on veut, par exemple, utiliser cette approche pour résoudre la loi générale de la dynamique $\vec{F} = m\vec{\gamma}$, il faut développer jusqu'au deuxième ordre car on connaît la valeur de la dérivée deuxième γ mais on ne connaît pas les autres dérivées. Newmark et Wilson ont approché la dérivée troisième $\dot{\vec{\gamma}}$ par $\frac{\gamma_{t+\tau}-\gamma_t}{\tau}$ pour obtenir le chemin implicite suivant:

$$\begin{aligned} \vec{V}_{t+\tau} &= \vec{V}_t + \tau\,((1-\alpha)\,\vec{\gamma}_t + \alpha\,\vec{\gamma}_{t+\tau}) \\ \vec{P}_{t+\tau} &= \vec{P}_t + \tau\,\vec{V}_t + \tfrac{\tau^2}{2}((1-\beta)\,\vec{\gamma}_t + \beta\,\vec{\gamma}_{t+\tau}) \end{aligned} \tag{3.3}$$

Cette méthode est inconditionnellement stable si $\alpha \geq \frac{1}{2}; \beta \geq \frac{1}{2}(\alpha + \frac{1}{2})^2$.

Les deux exemples précédents, donnent des chemins implicites, où l'application directe de ces deux méthodes ne donne pas directement la valeur de la fonction à l'instant t, mais elle donne un système d'équations à résoudre. Quand le système obtenu est linéaire, la résolution est directe. Par contre, quand il s'agit d'un système non linéaire ou discontinu, les méthodes implicites perdent leur avantage. En fait la résolution d'un système non linéaire n'est pas directe mais elle nécessite plusieurs itérations. De plus, bien que les méthodes implicites convergent, elles ne sont pas forcément plus précises que les méthodes explicites (figure 3.1). En fait l'erreur commise dépend aussi de la valeur de τ.

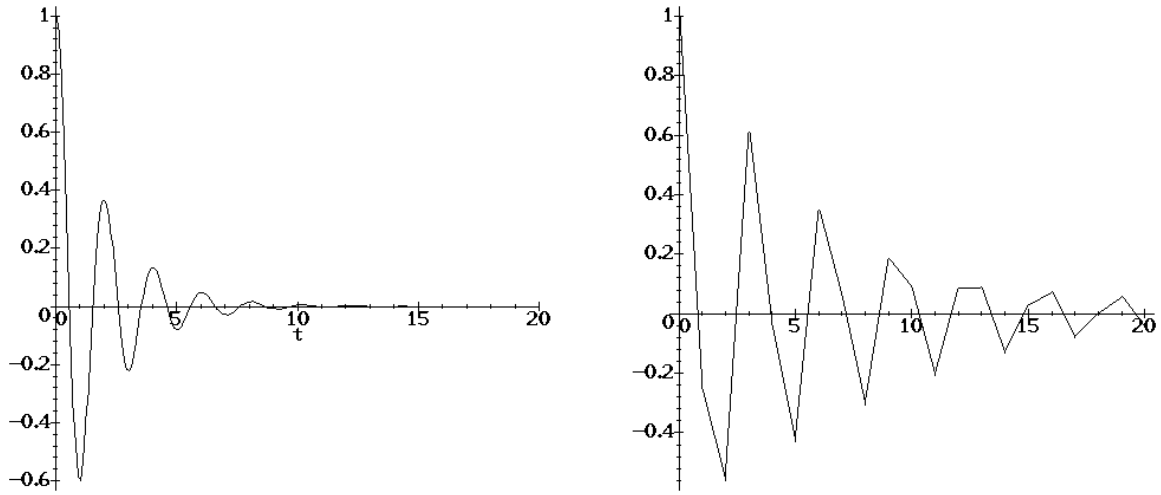


Figure 3.1: *Les méthodes implicites convergent à tous les coups mais elles ne donnent pas forcément une solution correcte. A gauche, la solution analytique et à droite la solution par la méthode de Newmark et Wilson.*

Le système (3.1) n'est *ni continu ni linéaire*, à cause des forces de collision et de contrôle qui peuvent être appliquées par l'utilisateur, à n'importe quel moment. Par

conséquent, il est difficile de résoudre ce système analytiquement, ou en utilisant une approche numérique implicite.

Pour résoudre le système, on a utilisé un chemin explicite du développement limité de deuxième ordre:

$$\begin{aligned}
\vec{V}_{t+\tau} &= \vec{V}_t + \tau\,\gamma_t + O(\tfrac{\tau^2}{2}\dot{\gamma}_t) \\
\vec{P}_{t+\tau} &= \vec{P}_t + \tau\,\vec{V}_t + \tfrac{\tau^2}{2}\,\gamma_t + O(\tfrac{\tau^3}{6}\dot{\gamma}_t) \\
\vec{W_{t+\tau}} &= \vec{W}_t + I^{-1}(\vec{N}_t - \vec{W}_t \wedge (I.\vec{W}_t))\tau + O(\tfrac{\tau^2}{2}\ddot{\vec{W}}_t)
\end{aligned} \tag{3.4}$$

L'avantage de cette approche explicite provient de son indépendance aux forces (c'est à dire qu'on n'a pas besoin de connaître à priori la valeur de la force pour la prendre en compte dans la résolution du système). Le système peut être donc utilisé comme une boîte noire sans avoir besoin de réécrire le système d'équations pour chaque exemple.

Cette approche a quand même quelques inconvénients:

- la solution obtenue est très sensible au pas de temps τ: si τ est très grand, cela peut causer une divergence numérique générée par le terme de l'erreur (figure 3.2); si τ est trop petit, le temps d'exécution est prohibitif. De plus, il est difficile de trouver le pas de temps qui vérifie la stabilité numérique, car ce pas de temps dépend de plusieurs facteurs comme les caractéristiques de la collision ou les propriétés de la déformation. En pratique, ce pas de temps est choisi d'une manière empirique après avoir fait plusieurs essais d'exécution du processus de simulation;

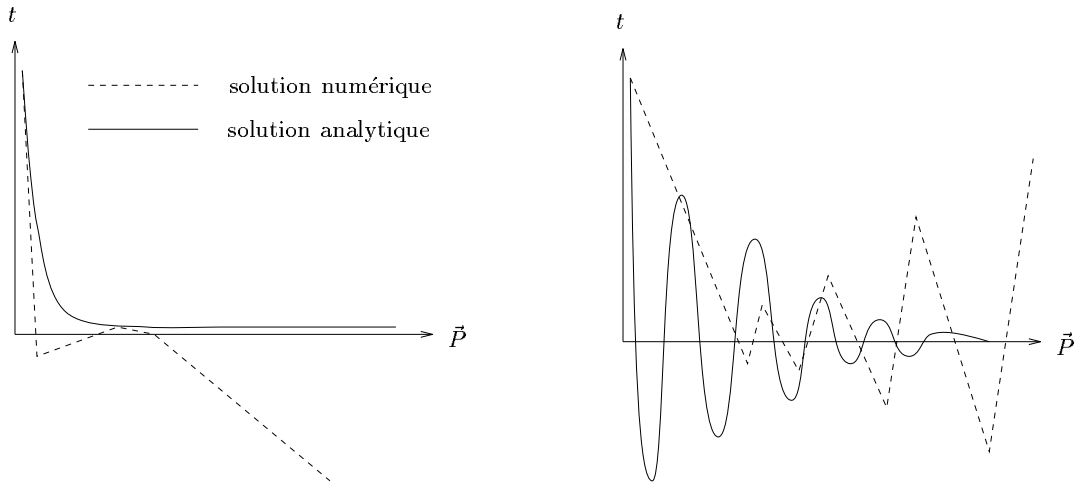


Figure 3.2: *Sans un pas de temps approprié la solution numérique peut diverger*

- A cause de la discontinuité du comportement, le terme de l'erreur ne peut pas être estimé par les méthodes classiques car on n'a pas une estimation des dérivées. Ceci a un inconvénient majeur pour des applications robotiques car on ne pourra pas prendre en compte l'incertitude.

3.4 Le pas de temps adaptatif et l'estimation de l'erreur

3.4.1 Problème et approche

Comme le montre l'équation (3.4), le terme de l'erreur associée au calcul de $\vec{V}_{t+\tau}$, $\vec{P}_{t+\tau}$ et $\vec{W}_{t+\tau}$ augmente quand τ et/ou la dérivée des accélérations ($\dot{\gamma}$ et $\ddot{\vec{W}}_t$) augmentent. Les valeurs de $\dot{\gamma}$ et $\ddot{\vec{W}}_t$ ne peuvent pas être contrôlées. Donc, le seul moyen pour contrôler l'erreur totale consiste à diminuer le pas de temps et par conséquent augmenter le temps d'exécution. Si le pas de temps est constant, la fréquence d'échantillonnage doit être deux fois plus grande que la plus haute fréquence dans la fonction $\vec{P}$ (théorie de Shannon [77]). Il suffit donc que la fréquence du signal augmente sur une petite période de temps pour que le temps d'exécution augmente énormément. Une stratégie efficace pour optimiser le temps d'exécution, tout en gardant une erreur constante, consiste à utiliser un pas de temps adaptatif τ (figure 3.3), qui varie en fonction de $|\dot{\gamma}|$ et de $|\ddot{\vec{W}}_t|$.

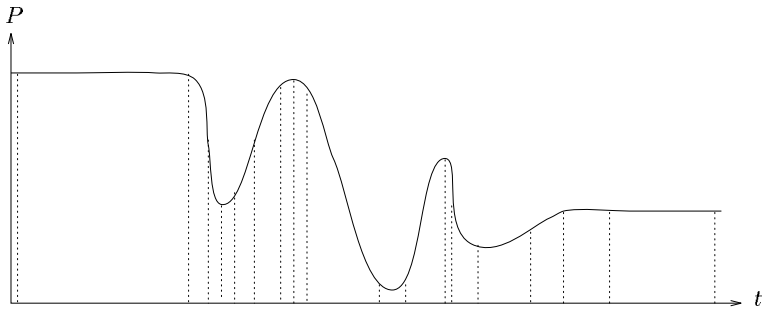


Figure 3.3: *Une bonne stratégie pour éviter la divergence numérique tout en réduisant le temps d'exécution, consiste à utiliser un pas de temps adaptatif τ qui permet d'intégrer le système en utilisant un nombre minimal d'échantillons.*

Le problème essentiel vient du fait que la valeur de $\dot{\gamma}$ n'est pas connu, et par conséquent elle ne peut pas être utilisée pour estimer le terme de l'erreur. De plus la discontinuité des forces extérieures, rend l'estimation de cette erreur impossible. Les intégrateurs numériques qui utilisent un pas de temps adaptatif tels que Runge-Kutta ou Adams-Moulton[83, 85] ne peuvent pas être appliquées dans notre cas, car elles demandent que la fonction soit continue. C'est pourquoi Baraff [4] a mis l'hypothèse que son objet est en mouvement libre entre deux collisions.

Afin de résoudre partiellement le problème, on a décidé d'estimer la valeur de l'erreur en évaluant ses conséquences sur la valeur de l'énergie mécanique [59, 62, 54] du système.

Soit E_m l'énergie mécanique d'un système physique S ($E_m = E_p + E_k$, où E_p est l'énergie potentielle et E_k est l'énergie cinétique). E_m dépend de $\vec{P}$ et de $\vec{V}$:

$$E_m = E_p + E_k = -\int_P^{P+\Delta P} \vec{F}.\delta\vec{P} + \frac{1}{2}m\Delta V^2$$

Trois cas doivent être considérés quand on raisonne sur la variation de E_m:

- Cas 1: S est un système isolé, sans dissipation de l'énergie. C'est le cas quand un objet (ou un ensemble d'objets) est libre (c'est à dire aucune force extérieure n'agit sur lui) et qu'il n'y a pas de force de frottement entre les différents objets qui constituent le système. Dans ce cas, E_m est constant (figure 3.4.a) et n'importe quelle variation de cette valeur est forcément due à l'augmentation de l'erreur numérique;

- Cas 2: S est un système isolé avec dissipation interne de l'énergie. C'est le cas quand il y a une force entre les différentes parties du système. Dans ce cas, E_m diminue (figure 3.4.b). La variation de l'énergie mécanique, pourtant naturelle, indique alors l'existence de perturbations dynamiques capable de faire augmenter le terme de l'erreur numérique;

- Cas 3: S n'est pas isolé. C'est le cas quand des forces extérieures agissent sur le système S (exemple. force de contrôle, force de collision, etc...). Dans ce cas, E_m peut augmenter (figure 3.4.c). Alors, la variation de l'énergie est, également, un témoin des perturbations dynamiques qui peuvent faire augmenter l'erreur numérique.

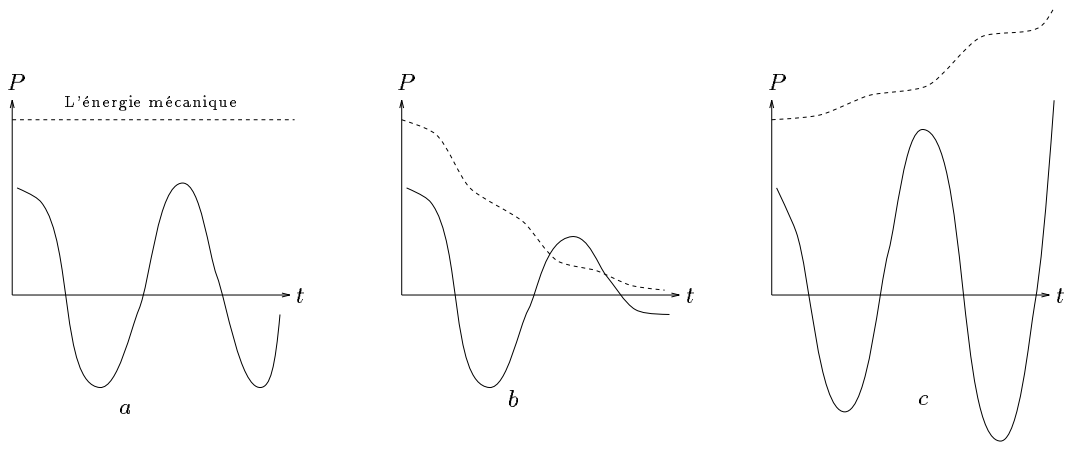


Figure 3.4: *(a) Un système isolé sans dissipation interne de l'énergie: E_m est constante. (b) Un système isolé avec dissipation interne de l'énergie: E_m diminue. (b) Un système non isolé: E_m peut augmenter.*

3.4.2 Détermination du pas de temps

Considérons tout d'abord le cas où l'énergie mécanique est constante, c'est à dire aucune force imprévue n'agit sur le système(figure 3.4.a). L'extension du domaine de validité, de l'algorithme sera discuté plus tard (§3.4.3). Comme nous l'avons déjà mentionné, l'énergie mécanique E_m est **constante** lorsque le système est isolé: $\Delta E_m = 0$. Si ΔE_m augmente pendant le traitement numérique, cela veut dire que cette variation "non physique" est le résultat de l'augmentation du terme d'erreur. Donc, l'idée de base consiste à évaluer la valeur de $|\Delta E_m|$ à chaque pas de temps et

à choisir le plus grand pas de temps qui satisfait la condition $|\Delta E_m| < \epsilon_e$, pour un ϵ_e donné par l'utilisateur. L'algorithme suivant donne une implantation pratique de ce résultat:

Soient $\vec{P_t}$, $\vec{V_t}$, $\vec{\Theta_t}$, $\vec{W_t}$ **et** $\vec{F_t}$
sont respectivement la position actuelle de l'objet,
sa vitesse, son orientation, sa vitesse de rotation,
et la somme des forces extérieures appliquées sur l'objet.
BOUCLE
  Calculer $\vec{P_{t+\tau}}$, $\vec{V_{t+\tau}}$, $\vec{\Theta_{t+\tau}}$ **et** $\vec{W_{t+\tau}}$
  Calculer ΔE.
  SI $|\Delta E| > \epsilon_e$ **ALORS**
    BOUCLE
      $\tau = \frac{\tau}{2}$
      Calculer $\vec{P_{t+\tau}}$, $\vec{V_{t+\tau}}$, $\vec{\Theta_{t+\tau}}$ **et** $\vec{W_{t+\tau}}$
      Calculer ΔE
    JUSQU'A $|\Delta E| < \epsilon_e$
  SINON
    $\tau = \frac{3}{2}\tau$
JUSQU'À la fin de la simulation

Justification: $|\Delta E_m| > \epsilon_e$ signifie que le terme d'erreur est très grand, et donc que le pas de temps τ est trop grand; $|\Delta E_m| < \epsilon_e$ signifie qu'il y a une forte chance que l'état du système soit "correct". Une justification intuitive de cette proposition est basée sur la continuité du mouvement de l'objet en question. Dans l'espace continu, une erreur correspond obligatoirement à une variation quelconque de la valeur de l'énergie mécanique. Si dans l'espace discret la valeur de l'énergie reste constante entre deux itérations, cela signifie que l'itération est correcte, ou que l'énergie est augmentée puis diminuée d'une valeur identique durant le pas de temps. Cela est peu probable, vue la continuité du mouvement et vu que le pas de temps est très petit (environ un dixième de seconde). Cela reste possible pour des grandes valeurs du pas de temps. Par exemple, quand deux objets s'interpénètrent, l'énergie mécanique augmente en continue jusqu'à ce que les deux objets se séparent de nouveau. Si le pas de temps était assez grand, pour que les deux objets puissent passer (en une seule itération) une distance plus grande que la somme de leurs dimensions, il serait alors possible que l'énergie soit constante, et que le mouvement ne soit cependant pas correct. En général on rencontre ces problèmes même si l'on utilise un pas de temps constant.
Dans la section suivante, on montre que l'approche reste applicable lorsque le système n'est pas isolé.

3.4.3 Traitement de perturbations externes

Quand une force externe F_e (une force de contrôle ou une force de collision) est appliquée sur le système physique, l'énergie mécanique E_m n'est plus constante

pendant un petit intervalle de temps. Après cette application, on peut prendre en compte l'énergie potentielle engendrée par F_e (en considérant qu'elle ne va pas changer dans le futur proche) et l'énergie mécanique va avoir une nouvelle valeur constante (figure 3.5) que l'on peut obtenir en ajoutant l'énergie potentielle engendrée par la nouvelle force. Le problème provient de la discontinuité de cette force et de la période de transition très petite. Pour que le passage entre les deux phases (avant l'application de la force et après cette application) soit fait sans erreur, il faut utiliser un pas de temps très petit ϵ_τ. Une bonne méthode pour choisir la valeur de ϵ_τ, tout en restant cohérent avec l'algorithme décrit ci-dessus, consiste à trouver la valeur τ pour laquelle ΔE_m entre t et $t+\tau$ soit plus petit que ϵ_e. Cela est toujours possible à cause de la **continuité** de E_m. Par conséquent, l'algorithme précédent est toujours applicable même quand le système physique n'est pas isolé.

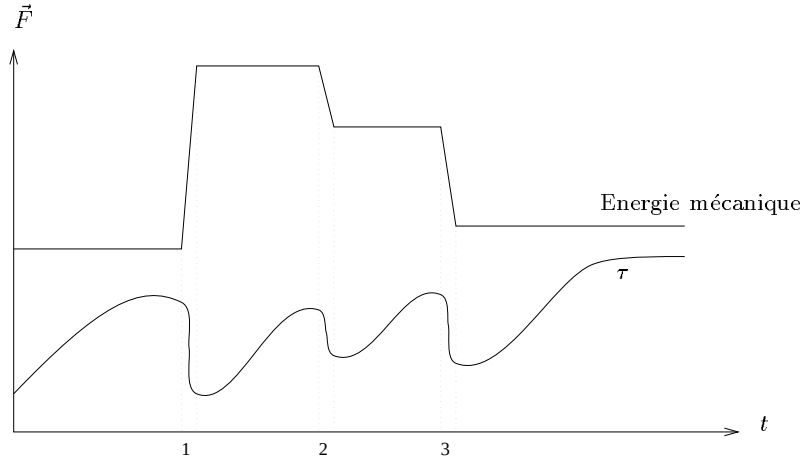


Figure 3.5: *Représentation schématique de la variation du pas de temps en fonction de la variation de l'énergie mécanique. Dans cette figure on voit trois perturbations externes. La valeur du pas de temps peut être saturé au bout d'un certain temps si l'objet est déformable. S'il est rigide et que les forces sont constantes, le pas de temps peut augmenter sans arrêt car la solution numérique dans ce cas devient analytique et le terme d'erreur s'annule.*

La validité de l'algorithme peut être, également, prolongé dans le cas ou la force F_e n'est pas dérivée d'un potentiel (exemple: force de frottement, ou $\int_P^{P+\Delta P} \vec{F}.\delta\vec{P} \neq 0$). En fait, l'algorithme est appliqué entre deux itérations, où les forces sont approximativement constantes (Cette hypothèse vient de l'utilisation d'une approche explicite). Donc la variation de la force du frottement peut être représenté par une fonction constante par morceaux (la figure 3.6 l'illustre).

La figure 3.7 montre l'évolution de τ quand une balle tombe sur le sol.

3.4.4 Caractéristiques principales de notre approche

Soit T_a, le temps d'exécution nécessaire pour achever une itération (passer de l'état à l'instant t à l'état à l'instant $t+\tau$) en utilisant un pas de temps adaptatif, T_c le temps d'exécution nécessaire pour achever une itération en utilisant un pas de

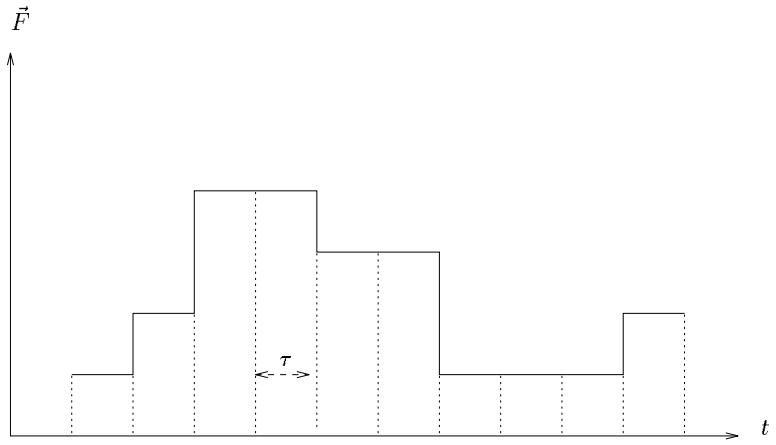


Figure 3.6: *En utilisant une approche explicite, on considère que la force est constante à chaque itération.*

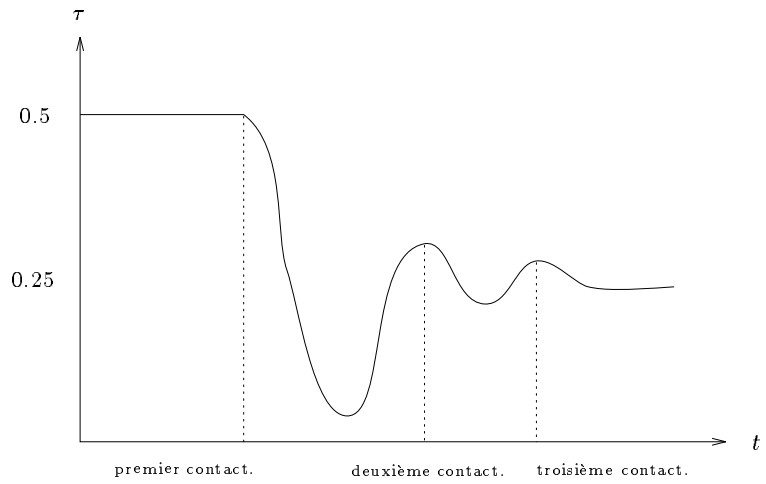


Figure 3.7: *L'évolution de τ quand une balle chute sur le sol. Remarquons que τ diminue quand la balle touche le sol et il augmente de nouveau quand la balle quitte le sol. Remarquons aussi, que la valeur de cette diminution dépend de la valeur de la force de la collision qui est plus importante quand la balle est plus rapide.*

temps constant, τ la valeur courante du pas de temps, τ_g le pas de temps qui vérifie la contrainte de l'énergie, et N_s le nombre d'itérations nécessaire pour atteindre τ_g.

- A chaque itération, on divise τ par 2 si celui-ci n'est pas satisfaisant. Donc, la valeur de N_s est trop petite. On peut écrire que:

$$\frac{\tau}{2^{N_s}} = \tau_g \Rightarrow 2^{N_s} = \frac{\tau}{\tau_g}$$

  donc,

$$N_s = \log_2 \frac{\tau}{\tau_g}$$

  Pratiquement et pour un grand nombre de tests, on a remarqué que $T_a \simeq 1.5 * T_c$. Cela peut être expliqué par la continuité du mouvement qui fait que τ_g et τ ont deux valeurs très proches $2\tau_g > \tau$. Alors $N_s = 1$ si $\tau_g < \tau$ et $N_s = 0$ sinon et la valeur moyenne de N_s devient $\frac{1+0}{2} = 0.5$.

- Le gain en temps d'exécution pendant t secondes est proportionnel au rapport entre le pas de temps moyen et le pas de temps minimal qui vérifie la contrainte de l'énergie:

$$Gain_t = \frac{\text{pas de temps moyen}}{\text{plus petit pas de temps}} > 1$$

  Expérimentalement, le gain était très important. Par exemple, il peut atteindre la valeur de 10000 pour une collision assez rigide.

- On n'utilise pas le même rapport quand on augmente le pas de temps (on multiplie par 1.5) et quand on diminue le pas de temps (on divise par 2), afin de ne pas tomber toujours sur les même valeurs et de pouvoir s'approcher autant que possible du plus grand pas de temps qui respecte notre contrainte.

- Pour estimer l'erreur commise en vitesse, on suppose que, dans le pire de cas, la variation de l'énergie ϵ_e est complètement transformée en énergie cinétique. Dans ce cas on peut écrire que:

$$\Delta E_k = \frac{1}{2} m \Delta V^2 \Rightarrow \varepsilon_v = \sqrt{\frac{2\varepsilon_e}{m}}$$

- Comme $\Delta P \simeq V\tau$, l'erreur commise en position est approximativement égale à: $\varepsilon_p \simeq \varepsilon_v * \tau$.

3.5 Expérimentations

L'exactitude de notre approche adaptative a été testée en utilisant MAPLE[2] sur des cas simples solvables analytiquement (deux particules liées par un ressort), on a comparé la solution obtenue en utilisant un pas de temps adaptatif PTA et celle obtenue en utilisant un pas de temps fixe PTF (figure 3.8).

[2] Maple est un logiciel de calcul algébrique, muni d'algorithmes numériques, symboliques et graphiques très efficaces

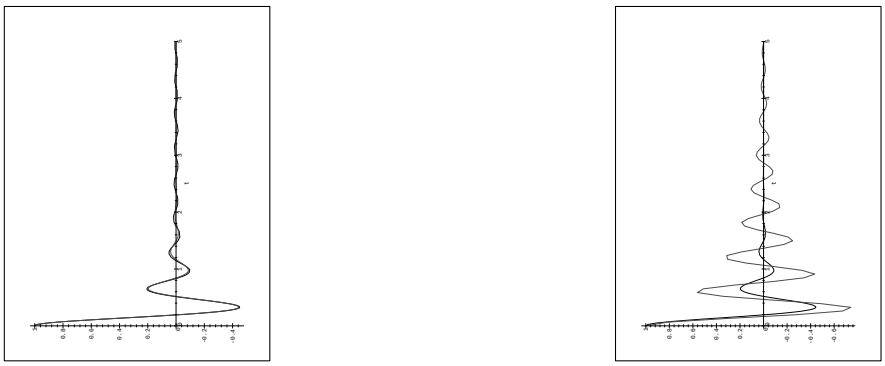


Figure 3.8: *Variation de la position relative de deux particules en fonction du temps. A gauche, on utilise un pas de temps adaptatif, l'erreur commise est de* 0.00006. *A droite on utilise un pas de temps constant. Les deux exemples tournent à la même vitesse car on a choisi le pas constant égal au pas de temps moyen utilisé par l'approche adaptative. L'erreur commise dans ce cas est égale à* 0.003. *Les paramètres dynamiques sont:* $\lambda = 100$, $\mu = 5$, $\tau_m = 0.065$ *et,* $\epsilon_e = 0.01$.

La solution obtenue par PTA, converge toujours beaucoup mieux que la solution obtenue par PTF, ainsi que l'erreur commise par PTA est plus petite que celle commise par PTF. Le gain, en temps d'exécution, obtenue en utilisant PTA est très variable, il dépend des caractéristiques physiques du phénomène à simuler. Pour le même exemple, la somme des valeurs d'erreurs en position sur un intervalle de temps, était proportionnelle à la valeur de ϵ_e (figure 3.9).

Pendant les tests les plus complexes que nous avons réalisé en utilisant *Robot*Φ, ce gain a varié entre 50 pour des cas très simples (un objet déformable qui bouge sous l'effet d'une force constante appliquée sur un de ses points), et 1000000 pour des cas plus compliqués (un robot qui entre en collision rigide avec un obstacle).

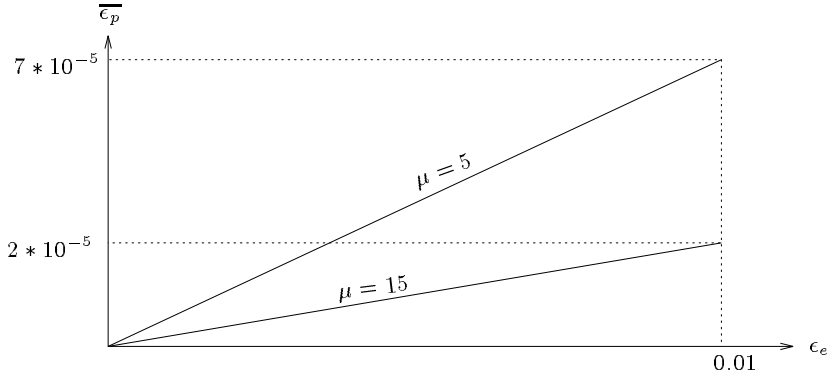


Figure 3.9: *Valeur de l'erreur commise en position* ϵ_p *en fonction de l'erreur commise en énergie* ϵ_e *pendant* 10 *secondes et pour deux valeurs différentes du facteur de la viscosité* μ. *Ce résultat est obtenu à l'aide de MAPLE.*

La figure 3.10 montre[3] un drap déformable sur lequel on a appliqué une force alternative perpendiculaire à sa surface et agissante sur son milieu. Le fait que la force change sa direction instantanément est une discontinuité qui implique la diminution du pas de temps pour minimiser l'erreur. Cette discontinuité s'est répétée 20 fois pendant le test. Le pas de temps moyen était d'environ 0.01 secondes tandis que le pas de temps minimal qui conserve les contraintes de la conservation de l'énergie était d'environ $10e-10$ secondes. Le temps d'exécution sur 20 secondes varie en fonction des paramètres entre 20 et 90 secondes sur un $INDIGO2$.

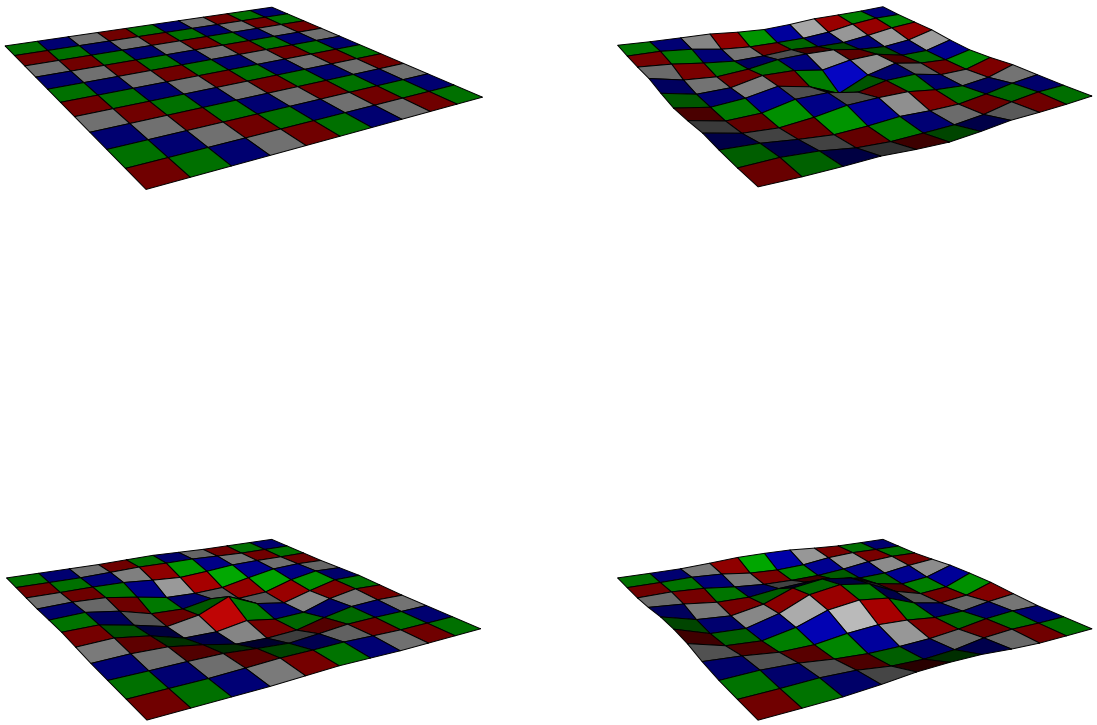


Figure 3.10: *Génération des ondes par une force alternative. Le gain était d'environ* 10^8

[3]le rendu sur l'écran donne des images lisses et réalistes. Le snapshot ne considère que la représentation interne des images

On peut obtenir une exécution en temps réel dans certains cas, mais ce n'est pas possible dans le cas général car on peut toujours fabriquer numériquement des fonctions de forces qui ne le permet pas.

Un autre problème à résoudre quand on utilise un pas de temps adaptatif concerne un type d'erreur qui n'a pas d'effet sur la valeur de l'énergie mécanique : Supposons deux balles B_1 et B_2 (figure 3.11) qui bougent en deux sens opposés avec une vitesse constante. Elles vont se toucher l'une l'autre après t_c secondes.

Parfois le système de simulation n'est pas capable de détecter cette collision. C'est le cas quand à l'instant t les deux balles étaient séparées et qu'à l'instant $t + \Delta t$ elles passent à une position symétrique sans se croiser. Cela peut arriver si la vitesse de ces deux balles leur permet de parcourir, en un seul pas de temps Δt, une distance plus grande que la somme de leurs rayons R. Pour éviter ce problème la valeur maximale du pas de temps doit être inférieure à $\frac{R}{V}$. Alors le nombre d'itérations minimales pour atteindre l'instant de la collision t est égal à $\frac{t_c}{\Delta t} = \frac{t_c * V}{R}$. Supposons qu'il nous faille s secondes pour accomplir une itération, le temps d'exécution nécessaire pour atteindre t est égal à $Exe_t = \frac{s * t_c * V}{R}$. Alors la condition nécessaire pour obtenir un temps réel est $Exe_t <= t_c \Rightarrow \frac{s*V}{R} < 1 \Rightarrow s * V < R$, qui ne peut pas être toujours vraie car elle dépend de la vitesse de l'objet. Pour surmonter ce problème, il faut ajouter d'autres tests. Par exemple il faut choisir le pas de temps tel que le mouvement d'un objet pendant τ soit inférieur à sa distance minimale aux autres objets.

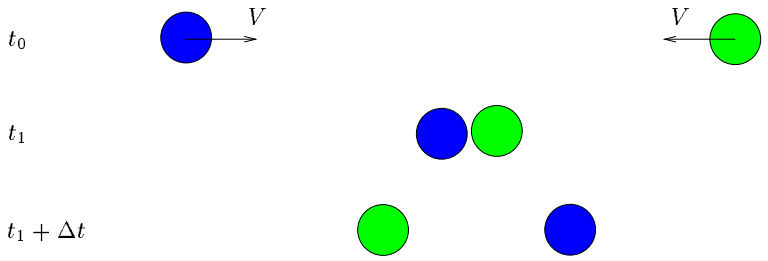


Figure 3.11: *Pour être sur que nous allons détecter le contact entre ces deux balles, il faut que le pas de temps vérifie que* $Exe_t <= t_c \Rightarrow \frac{s*V}{R} < 1 \Rightarrow s * V < R$. *Cela ne peut pas être toujours vrai, pour cela il faut introduire d'autres tests de proximité.*

3.6 Conclusion

Le mouvement et les déformations d'un objet physique sont les résultats des mouvements de ses primitives élémentaires. Les mouvements de ces primitives sont caractérisés par un système d'équations différentielles discontinu et non-linéaire. La résolution de ce système nécessite parfois une fréquence d'échantillonnage très élevé, afin de minimiser l'erreur commise et d'éviter la divergence numérique. Cela diminue énormément la performance du système dynamique. Une approche adaptative, basée sur la notion de l'énergie mécanique, a été développée afin d'évaluer l'erreur et de la minimiser tout en maximisant le pas d'échantillonnage afin d'augmenter la

performance du système. Cette approche consiste à diminuer le pas de temps quand on détecte une variation significative de l'énergie mécanique et à l'augmenter dans le cas contraire. En fait, l'énergie mécanique est très sensible aux différents types de perturbations qui peuvent affecter la dynamique d'un système et dont le traitement nécessite une fréquence d'échantillonnage très élevée. La valeur de cette énergie est constante quand le système est libre: elle change si le pas de temps est inapproprié. En effet elle augmente si deux objets entrent en collision, et elle diminue si deux objets se frottent. C'est pourquoi l'énergie mécanique est un très bon diagnostic pour détecter l'erreur dans les méthodes numériques.

Chapter 4

Interactions

4.1 Introduction

Quand deux objets bougent dans le même environnement, ils risquent de rentrer en contact. Cela implique deux problèmes: d'abord comment détecter le contact entre les deux objets, et puis comment calculer les forces résultantes de ce contact ?

La détection des contacts entre objets en mouvement est la partie la plus critique pour la plupart des algorithmes géométriques, et en particulier pour les algorithmes physiques (la partie la plus importante du temps d'exécution est consacré à détecter les contacts entre les différents objets). Des résultats intéressants ont été obtenus pour la détection de contacts entre des objets rigides et convexes. Dans notre cas, il n'est pas possible de faire l'hypothèse qu'un objet est convexe car cet objet est déjà déformable et il peut passer dynamiquement à l'état concave, même si initialement il était convexe. De plus les objets peuvent, fictivement, s'interpénétrer pour deux raisons: (1) on utilise la méthode de la pénalité pour calculer la force de la collision (§4.3). Cette méthode approche les déformations locales des objets lors de la collision par l'interpénétration maximale; (2) on utilise un pas de temps adaptatif pour optimiser le temps de calcul, et il se peut que les deux objets s'interpénètrent largement en premier temps avant que le système détecte la violation de l'énergie mécanique et répète l'itération pour une valeur de pas de temps plus petite. Il nous faut donc un algorithme efficace (linéaire) capable de détecter le contact entre deux polyèdres déformables qui peuvent s'interpénétrer, fictivement, au cours de leur mouvement.

Quand le contact a lieu entre deux objets, deux types de force apparaissent; la collision qui se représente par une force répulsive dont le rôle est de séparer les deux objets, et le frottement qui est une force tangente au plan de contact dont le rôle est de ralentir la vitesse relative de l'un par rapport de l'autre. De plus, le milieu dans lequel l'objet évolue, s'il n'est pas le vide, exerce une force qui ralentit sa vitesse absolue. On appelle cette force la force de la viscosité du milieu.

Ce chapitre explique comment on peut détecter et localiser efficacement (en temps linéaire) le contact entre deux polyèdres déformables, comment calculer les trois forces d'interaction (collision, frottement et viscosité) et les intégrer ensemble.

4.2 La détection du contact

Plusieurs résultats intéressants sont présentés dans la littérature pour calculer la distance et détecter le contact entre des polyèdres rigides.

- Lin & Canny [69] ont proposé un algorithme linéaire $O(n)$ (où n est le nombre de caractéristiques des deux polyèdres, une caractéristique peut être une facette, une arête ou un sommet) qui calcule la distance entre deux polyèdres rigides et convexes. Cet algorithme peut profiter de la continuité du mouvement pour réduire sa complexité à $O(1)$, c'est-à-dire que la détection se fait en temps constant. Cet algorithme ne donne que la distance positive [1] entre les deux polyèdres. Il a été étendu par [89] afin de localiser le contact entre les deux polyèdres, lorsqu'ils s'interpénètrent;

- Gilbert et al [39] ont proposé un algorithme capable de calculer en temps quadratique $O(n^2)$ (où n est le nombre de sommets des deux polyèdres), la distance positive entre les enveloppes convexes de deux ensembles de points (sans calculer leurs enveloppes convexes), et de donner une approximation de la distance négative [2] pour des petites interpénétrations. Cet algorithme peut profiter de la continuité du mouvement pour réduire sa complexité à $O(n)$;

- Garcia-Alonso et al [33], ont proposé un algorithme qui représente un objet par sa boîte min-max, par son container et par voxels. Ces techniques sont très intéressantes pour optimiser le calcul de la distance quand les objets sont séparés, il devient moins efficace quand ils sont en contact et il ne permet pas de localiser les différents points de contact quand les objets sont déformables;

Pour des polyèdres concaves, on peut les diviser en plusieurs polyèdres convexes et appliquer l'un des algorithmes précédents pour détecter les contacts entre eux. Le problème devient plus difficile quand on traite des objets déformables, car ils basculent dynamiquement, pendant le mouvement, entre les deux états convexe et concave. De plus, la décomposition dynamique en polyèdres convexes, quand c'est possible, nécessite un temps d'exécution très important.

- Baraff & Witkin [9] divisent statiquement chaque objet en plusieurs sous-objets dont les déformations, pendant le mouvement, sont polynomiales de premier degré. Cela garantie que les sous-objets ne passeront pas à l'état concave pendant leurs déformations. Dans ce cas, on peut utiliser un des algorithmes précédents pour détecter le contact entre deux sous-objets. Le problème est dû au nombre des sous-objets qui peut être très grand si on veut modéliser un objet très déformable;

- Volino & Thalmann [95] proposent un algorithme hiérarchique qui détecte le contact de différentes parties du même objet, mais ce n'est pas notre cas.

[1] La distance est dite positive si les deux polyèdres sont séparés l'un de l'autre
[2] La distance est dite négative si les deux polyèdres s'interpénètrent

4.2.1 Description générale de l'approche

Etant donné deux polyèdres A et B, A est composé de N_A facettes et N_a sommets, B est composé de N_B facettes et N_b sommets. Soit C_X l'enveloppe convexe du polyèdre X et F_X^i la $i^{\text{ième}}$ facette du polyèdre X. Une approche de base consiste à détecter la collision entre chaque paires de facettes (F_A^i, F_B^j). Cela nécessite $N_A * N_B$ opérations, et un temps d'exécution relativement long. Notre approche [64] peut optimiser le temps de calcul, en éliminant, en premier lieu, sur chaque polyèdre les facettes qui ne peuvent pas être en contact. Pour éliminer ces facettes on commence par calculer la zone d'interpénétration entre les deux enveloppes convexes C_A et C_B. Cette zone est approchée par un cylindre qui le contient. Les facettes qui sont à l'extérieur de cette zone ne peuvent pas être en contact. On peut en temps constant vérifier si une facette est à l'extérieur d'un cylindre, donc l'élimination des facettes peut être faite en temps linaire $O(N_A + N_B)$.

Pour trouver la zone d'interpénétration, on a besoin de certains paramètres de contact. Ces paramètres peuvent être obtenus par extension de l'algorithme de Gilbert (§4.2) afin de calculer la valeur de la distance négative.

Après avoir éliminé les facettes qui ne peuvent pas être en contact, notre algorithme retourne deux ensembles de facettes qui peuvent être en collision, ainsi que la distance négative entre les deux enveloppes convexes C_A et C_B et la direction du contact.

4.2.2 Elimination des facettes

L'élimination des facettes, nécessite une approximation de la zone d'interpénétration entre les deux enveloppes convexes des deux polyèdres. Le calcul de cette zone est basé sur l'utilisation de quelques paramètres dits "paramètres de contact" (voir figure 4.1) :

Déf1: On appelle distance négative d_n, le plus petit déplacement qui permet de séparer les deux enveloppes convexes C_A et C_B.

Déf2: On appelle direction du contact $\vec{n}_{A/B}$, la direction dans laquelle le plus petit déplacement doit être accompli. Elle est dirigée de A vers B.

Déf3: On appelle plan du contact $P_{A/B}$, le plan dont le vecteur directeur est $\vec{n}_{A/B}$ et qui passe par le point le plus proche de C_A à C_B.

Déf4: On appelle zone d'impact $Z_{A/B}$, l'enveloppe convexe de la projection de la partie de A qui se trouve derrière le plan $P_{B/A}$ (donc qui pénètre effectivement le polyèdre B), sur le plan $P_{B/A}$.

Etant donné les paramètres du contact, on peut trouver une boîte englobante de la zone d'interpénétration. Cette boîte est limitée par les deux plans de contact $P_{A/B}$ et $P_{B/A}$. De plus elle est limitée par l'intersection des deux zones d'impact $Z_{A/B}$ et $Z_{B/A}$. Cette intersection sera approchée par un cercle pour pouvoir calculer en temps linéaire les zones d'impact et pour simplifier la forme de la boîte englobante. Pour déterminer rapidement si une facette est à l'extérieur de cette zone, on peut appliquer deux tests:

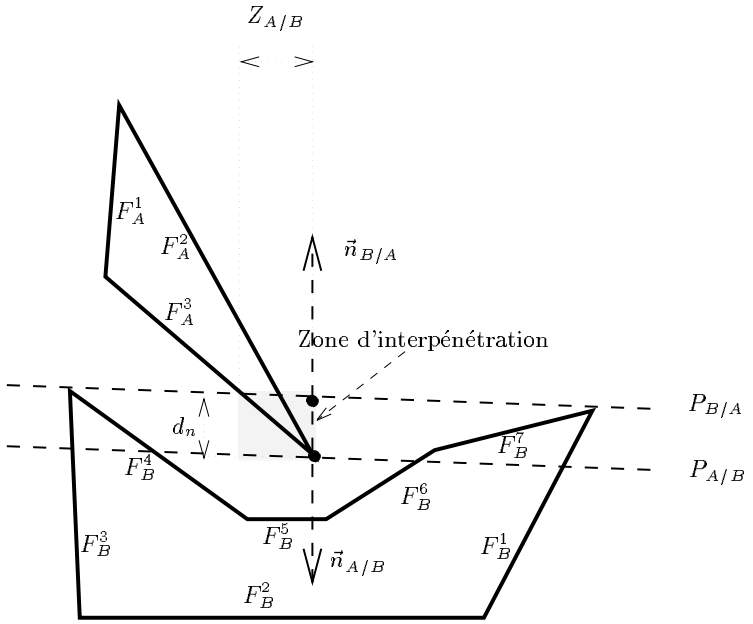


Figure 4.1: *Les facettes F_A^1, F_A^2, F_B^1, F_B^2, F_B^3, F_A^1, F_B^2, F_B^5, F_B^1, F_B^3, F_B^6, F_B^7 ne peuvent pas être en contact car elles sont à l'extérieur de la zone d'interpénétration. Seule F_A^2, F_A^3 et F_B^4 peuvent être en contact.*

- si $Distance(F_A^i, P_{B/A}) > 0$, alors la facette ne peut pas être à l'intérieur de la zone d'interpénétration entre C_A et C_B et elle peut donc être éliminée. C'est le cas des facettes F_A^1, F_B^2, F_B^5;

- si la projection de F_A^i sur $P_{A/B}$ se trouve à l'extérieure de $Z_{B/A}$, alors la facette est à l'extérieur de la zone d'interpénétration et elle peut donc être éliminée. C'est le cas des facettes F_B^1, F_B^3, F_B^6, F_B^7.

Les deux derniers critères peuvent détecter rapidement si une facette est à l'extérieur de la zone d'interpénétration mais pas l'inverse. Par exemple la facette $\{F_B^4\}$ n'a pas été éliminée. L'utilisation de ces derniers tests à la place d'un algorithme qui calcule la distance exacte entre une facette et un cylindre, est justifiée par le fait qu'on veut des tests rapides. Pratiquement, ces deux tests éliminent la majorité des facettes qui sont à l'extérieures de la zone d'interpénétration.

Après avoir appliqué ces tests sur chaque facette, l'algorithme retourne deux ensembles de facettes qui n'ont pas été éliminées (Dans notre exemple, sont $\{F_A^3, F_A^2\}$ et $\{F_B^4\}$). Ces tests sont atomiques, et ils peuvent être fait en temps constant pour chaque facette. Alors le temps nécessaire pour accomplir cette élimination dépend, seulement, du nombre de facettes $N_A + N_B$. Par conséquent, pour obtenir une complexité linaire de notre algorithme, il faut pouvoir obtenir les paramètres en temps linéaire.

4.2.3 Détermination des paramètres du contact

L'algorithme de Gilbert [39] est dessiné pour calculer la distance Euclidienne entre les deux enveloppes convexes de deux polyèdres A et B (sans calculer leurs enveloppes convexes) caractérisés par leurs sommets N_a et N_b dans un espace de m dimensions. La distance Euclidienne entre C_A et C_B est défini par:

$$\begin{aligned} d(C_A, C_B) &= \min\{|x-y| : x \in C_A, y \in C_B\} \\ &= \min\{|z| : z \in C_A \ominus C_B\} \\ C_A \ominus C_B &= \{z : z = x - y, x \in C_A, y \in C_B\} \end{aligned}$$

Alors, la distance entre C_A et C_B est égale à la distance entre l'origine O et leur différence de Minkowski $C_A \ominus C_B$ (La différence de Minkowski de deux polyèdres convexes est toujours un polyèdre convexe). L'algorithme de Gilbert trouve en temps linéaire le point $\vec{P} = \vec{P}_A - \vec{P}_B$ de $C_A \ominus C_B$ le plus proche de l'origine et il retourne $\vec{P}_A$ et $\vec{P}_B$. Donc, $d = |P_A - P_B|$ et $\vec{n}_{A/B} = \frac{\vec{P}_B - \vec{P}_A}{|\vec{P}_B - \vec{P}_A|}$. Bien que cet algorithme soit basé sur la notion de la différence de Minkowiski $C_A \ominus C_B$, il ne calcule pas explicitement cette différence. C'est pourquoi cet algorithme peut profiter de la continuité du mouvement de A et B pour converger en temps linéaire $O(N_a + N_b)$.

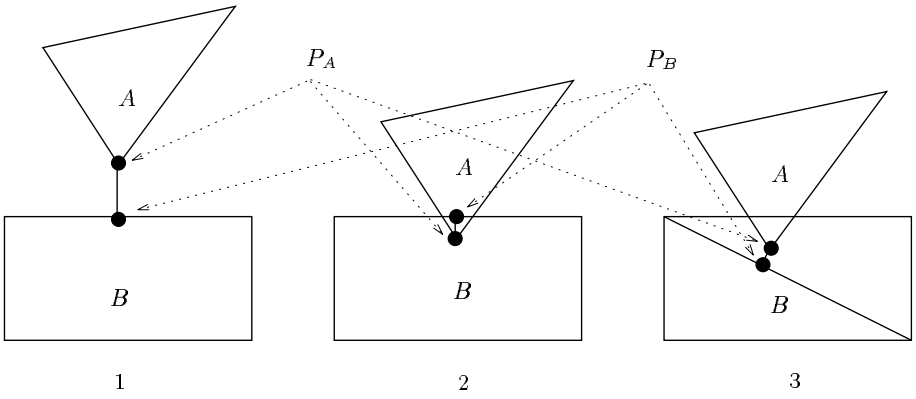


Figure 4.2: *L'algorithme de Gilbert donne la distance entre deux polyèdres et il retourne les deux points les plus proches P_A, P_B (1). Lorsque les deux polyèdres s'interpénètrent (2), l'algorithme peut ne pas donner une distance négative correcte (3), car il est basé sur un test local de distance.*

Quand les deux polyèdres s'interpénètrent faiblement, l'algorithme détecte cette situation et donne une approximation (voir figure 4.2) de la distance négative (ceci ne marche que si l'interpénétration est très faible). Comme l'algorithme considère les enveloppes convexes des deux polyèdres, il se peut que la distance soit négative alors que les deux polyèdres ne sont pas réellement en contact. C'est pourquoi il n'est pas possible d'utiliser directement cet algorithme pour résoudre le problème de la collision entre objets flexibles. Dans la suite, on montrera comment cet algorithme peut être utilisé pour trouver les paramètres du contact.

Détermination de la "direction du contact"

On peut distinguer deux cas:

Cas1: $d(C_A, C_B) > 0$. Dans ce cas $\vec{n}_{A/B} = \frac{\vec{P}_B - \vec{P}_A}{|\vec{P}_B - \vec{P}_A|}$, où P_A et P_B sont les deux points les plus proche donnés par l'algorithme de Gilbert.

Cas2: $d(C_A, C_B) < 0$. Dans ce cas les deux enveloppes convexes s'interpénètrent, et l'algorithme de Gilbert ne peut plus être utilisé pour calculer P_A et P_B. De plus, $\vec{n}_{A/B}$ est la direction dans laquelle on doit faire bouger C_A et C_B afin de les séparer d'un déplacement minimal.

L'idée de l'algorithme est basée sur les propriétés suivantes de $\vec{n}_{A/B}$ quand les deux enveloppes convexes s'interpénètrent. Pour des raisons de simplicité, les propriétés sont justifiées dans l'espace de la différence de Minkowski, bien qu'en pratique, l'on ne calcule pas cette différence. On dénote par $\vec{n}$ la direction du contact entre l'origine et $C_A \ominus C_B$. $\vec{n}$ est la direction dans laquelle on doit déplacer l'origine O afin de le faire sortir de $C_A \ominus C_B$ avec un déplacement minimal.

Propriété 1: La direction du contact obtenue après un déplacement dans la direction $\vec{n}_{A/B}$ est toujours $\vec{n}_{A/B}$.

Justification: Comme $C_A \ominus C_B$ est un polyèdre convexe, la partie de $C_A \ominus C_B$ la plus proche de l'origine O (l'origine O est à l'intérieur de $C_A \ominus C_B$ car C_A et C_B s'interpénètrent) est une facette. Alors $\vec{n}$ est perpendiculaire à cette facette. Quand O bouge dans la direction de la normale sur cette facette, cette facette reste la partie de $C_a \ominus C_b$ la plus proche de O (figure 4.3).

Propriété 2 : il y a une zone compacte de vecteurs $z = \vec{z}_i$ autour de $\vec{n_{a/b}}$ vérifiant que pour chaque déplacement selon $\vec{z}_i \in z$ (qui sépare C_a et C_b) la nouvelle direction du contact (qui est positive et obtenue par l'application de l'algorithme du Gilbert) est plus proche de $\vec{n_{a/b}}$ que $\vec{z}_i$ (figure 4.3).

Justification : comme $C_a \ominus C_b$ est un polyèdre convexe et O appartient à son intérieur, seul le type de contact point/facette est possible. Ce type de contact, entre l'origine et un polyèdre, correspond à trois types de contact entre deux polyèdres, facette/facette, facette/arête, et facette/sommet. Donc, la partie la plus proche de O est une facette f. La figure 4.3 montre que si on déplace O hors de $C_A \ominus C_B$ selon la direction $\vec{z}_i$ en coupant la facette f, la nouvelle direction du contact est plus proche de $\vec{n}$ (qui est normale à la facette) que $\vec{z}_i$, car α est un angle obtus qui varie entre 180^o (quand $\vec{z}_i$ passe par une arête) et $90°$ (quand la nouvelle direction du contact est perpendiculaire à f). La zone Z est donc définie par l'ensemble de vecteurs z_i qui passent par la facette la plus proche de O.

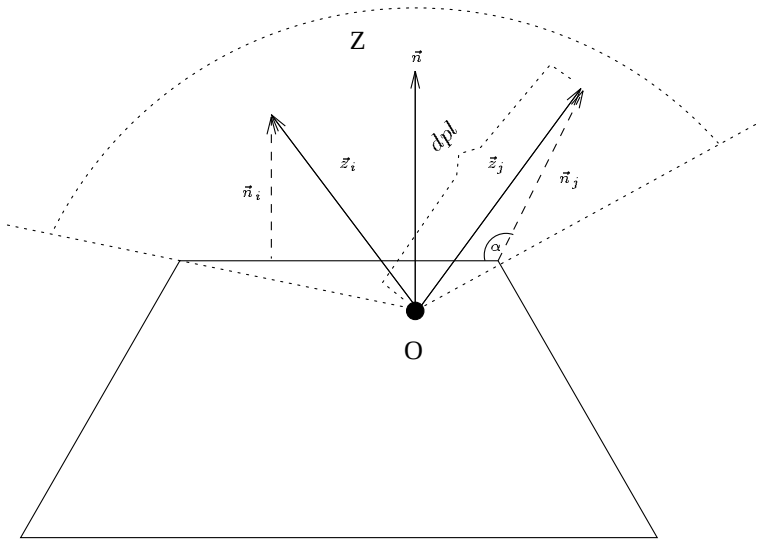


Figure 4.3: *A l'intérieur de la zone Z, n'importe quel déplacement ($\vec{z}_i$,$\vec{z}_j$,..) de O va donner une nouvelle direction de contact ($\vec{n}_i$,$\vec{n}_j$,..) qui est plus proche de la vraie valeur de la direction du contact $\vec{n}$.*

Etant donnée une valeur initiale $\vec{n}_0 \in Z$ de la direction du contact, les deux propriétés, mentionnées ci-dessus, nous permettent de trouver $\vec{n}_{A/B}$ en utilisant l'algorithme suivant:

Trouver_La_Direction_Du_Contact($\vec{n}_0$)
{
Séparer C_A et C_B selon la direction $\vec{n}^0$.
Appliquer l'algorithme de Gilbert afin d'obtenir
une nouvelle valeur de la direction du contact $\vec{n}_{A/B}$.
SI $\vec{n}_0 = \vec{n}_{A/B}$ ALORS
RETOURNER $\vec{n}_{A/B}$
SINON
Trouver_La_Direction_Du_Contact($\vec{n}_{A/B}$)
}

Remarque: Comme la direction du contact (quand la distance est négative) est forcément perpendiculaire à une facette, on pourrait trouver en cherchant la facette la plus proche de l'origine, mais cela nécessiterait un temps de calcul quadratique $O(N_a * N_b)$ car $C_A \ominus C_B$ contient $N_a * N_b$ sommets. Or notre but est d'obtenir un algorithme linéaire. De plus, les facettes de $C_A \ominus C_B$ ne sont pas données explicitement, car $C_A \ominus C_B$ est seulement défini par ses sommets.

Notre algorithme fait plusieurs itérations pour déterminer la direction du contact. A chaque itération i, la valeur de $\vec{n}_0^i$ est plus proche de $\vec{n}_{A/B}$ que $\vec{n}_0^{i-1}$. Donc, l'algorithme est convergant. La complexité de cet algorithme est I fois la complexité de l'algorithme de Gilbert, où I est le nombre d'itérations nécessaires pour converger. Donc, la complexité de l'algorithme est $O(I.(N_a + N_b))$. La figure 4.4 montre un exemple.

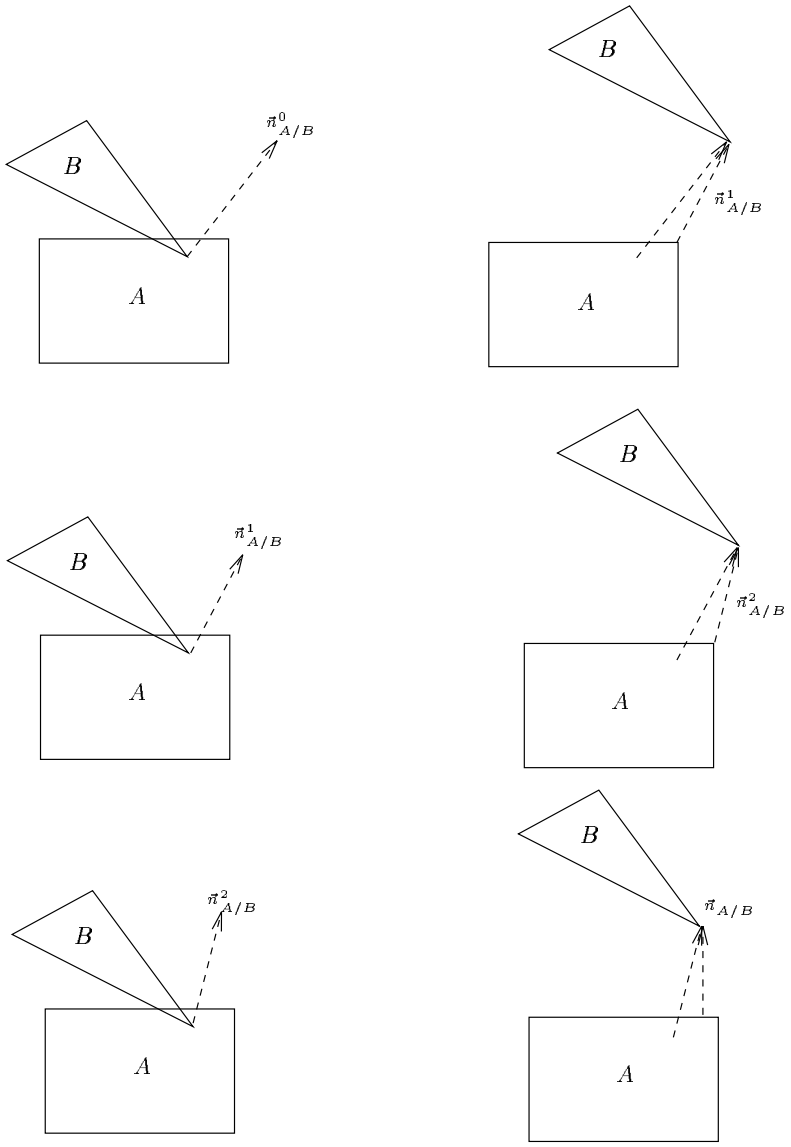


Figure 4.4: *Pour trouver $\vec{n}^{i+1}_{A/B}$, on déplace A par rapport à B selon la direction $\vec{n}^i_{A/B}$ de manière que leurs enveloppes convexes soient séparées, puis l'application de l'algorithme de Gilbert donne $\vec{n}^{i+1}_{A/B}$. Dans cet exemple $\vec{n}^3_{A/B}$ est égal à $\vec{n}_{A/B}$*

Le comportement de cet algorithme dépend de deux paramètres $\vec{n}_0$ et *dpl*:

choix de $\vec{n}_0$: à cause de la continuité du mouvement, la direction du contact à l'instant t peut être utilisée comme une valeur initiale de la direction du contact à l'instant $t+\tau$. En pratique, la distance initiale entre les deux polyèdres avant le contact est positive. Cette distance positive (donnée par l'algorithme de Gilbert) peut donc être utilisée pour initialiser la direction du contact lorsqu'on détecte une distance négative. Notre algorithme converge toujours vers la facette par laquelle passe la demi-ligne qui commence par O et qui est dirigée par $\vec{n}_0$. Cela rend l'algorithme robuste même pour des valeurs significatives d'interpénétration. La figure 4.5 montre ce qui se passe quand les deux polyèdres entrent en contact. D'abord quand ils sont séparés, l'origine est à l'extérieur de $C_A \ominus C_B$ et la direction du contact est $\vec{k_0}$. Quand les deux polyèdres rentrent en contact, l'origine devient à l'intérieur de $C_A \ominus C_B$. Le fait que l'algorithme utilise $\vec{k}_0$ comme valeur initiale de la direction de contact, la fait converger vers $\vec{k}_0$ quelque soit le montant de l'interpénétration. Donc $\vec{k}_0 = \vec{k}_1 = \vec{k}_2 = \vec{k}_3$. Cela ne veut pas dire qu'après le contact, la direction du contact devient constante: la direction du contact est insensible aux déplacements en profondeur, par contre elle est sensible aux déplacements tangentiels à la surface. La figure 4.5 montre l'évolution de la direction de contact (dans l'espace de $C_A \ominus C_B$) quand les deux objets se déplacent l'un par rapport à l'autre tout en restant en contact;

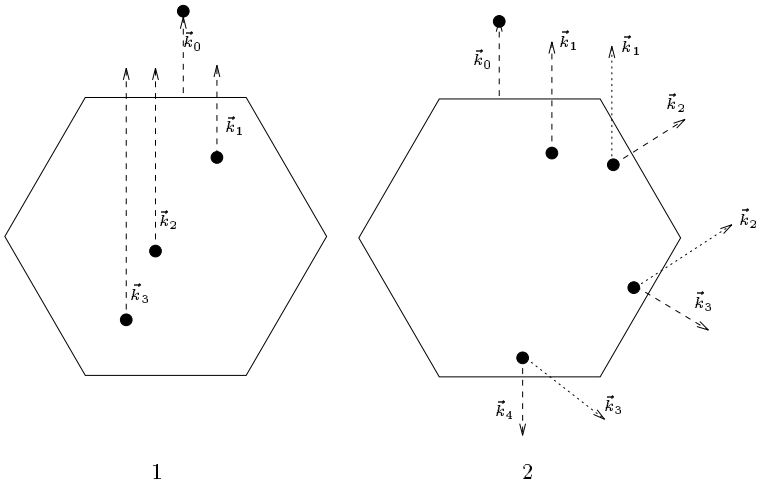


Figure 4.5: *(1) La direction du contact obtenue par notre algorithme est insensible au montant de l'interpénétration (2) mais il est sensible aux déplacements tangentiels*

choix de *dpl*: *dpl* est par définition la plus petite valeur de déplacement qui permet de séparer les deux polyèdres. Plus le déplacement est petit, plus la direction de contact, obtenue après ce déplacement, est plus proche de la valeur réelle de la direction du contact (voir figure 4.6). Il faut donc éviter de trop éloigner A de B et essayer de trouver la valeur minimale capable de les séparer. Nous avons décidé de choisir pour cela une valeur proportionnelle à celle de la distance négative obtenue à l'itération précédente, puisque cette valeur a maintenant

une signification réelle. Si cette valeur est trop petite, l'algorithme de Gilbert détectera une distance négative, ce qui nous conduira à répéter récursivement le processus d'éloignement pour une valeur plus petite de dpl.

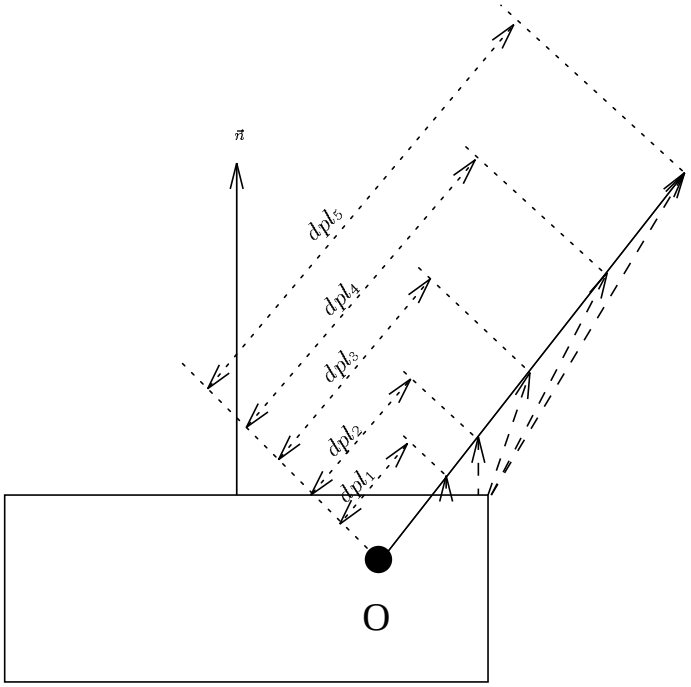


Figure 4.6: *Plus le déplacement est petit, plus la direction de contact, obtenue après ce déplacement, est plus proche de la valeur réelle de la direction du contact*

Ces choix de $\vec{n}^0_{A/B}$ et de dpl, nous permettent de converger très vite. En pratique, la valeur de I est très proche de 1.

Détermination de la valeur de la "distance négative" d_n

Une fois que la direction du contact $\vec{n}_{B/A}(t)$ est déterminée, on peut facilement trouver la valeur de la distance négative $d_n(t)$. Pour cela on applique un déplacement dpl sur le polyèdre A selon la direction $\vec{n}_{B/A}(t)$, puis on applique l'algorithme de Gilbert afin d'obtenir une distance positive d_p. Alors la valeur de la distance négative d_n peut être calculée par la relation $d_n = dpl - d_p$. La valeur de dpl dans ce cas est la dernière valeur obtenue pendant le processus de la détermination de la direction du contact et qui permet de séparer les deux enveloppes convexes des deux polyèdres.

Détermination de la "zone d'impact" $Z_{A/B}$

La zone d'impact $Z_{A/B}$ est par définition, l'enveloppe convexe de la projection de la partie de A qui se trouve derrière le plan $P_{B/A}$ (donc qui interpénètre réellement le polyèdre B), sur le plan $P_{B/A}$. Il n'est pas possible de trouver en temps linéaire la projection exacte de A sur $P_{B/A}$. C'est pourquoi on surestime cette projection par

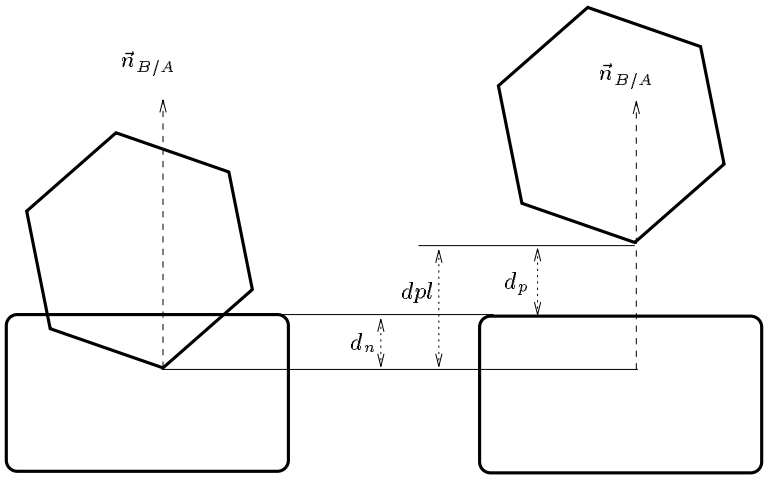


Figure 4.7: *Etant donné la direction de contact $\vec{n}_{B/A}$, la distance négative est égale à la différence entre la valeur du déplacement dpl et la valeur de la distance positive obtenue après le déplacement d_p: $d_n = dpl - d_p$*

l'enveloppe convexe des projections de tous les sommets de A (qui se trouve derrière $P_{B/A}$) et tous les points d'intersections entre une arête de A et $P_{B/A}$ (voir figure 4.8). On peut trouver ces projections et leur enveloppe convexe en temps linéaire $O(N_a + N_A)$, mais pour tester si une facette F_B^i est à l'intérieur ou à l'extérieur de $Z_{A/B}$, il nous faut un temps linéaire $O(n)$ où n est la taille de cette enveloppe convexe qui est du même ordre de grandeur que N_a. Pour N_B facettes, on a, donc, besoin de $N_B * N_a$ opérations pour éliminer les facettes. Pour réduire davantage la complexité de l'algorithme, on surestime la zone d'impact par le plus petit cercle qui contient tous les sommets de cette enveloppe convexe. Le centre C de ce cercle se trouve au centre géométrique de ces sommets, et le rayon R de ce cercle est la plus grande distance entre le centre géométrique et ces sommets. Pour qu'une facette soit à l'intérieur de la zone d'impact, il suffit qu'un de ses trois sommets (Les facettes sont triangulaires) soit à l'intérieur du cercle qui représente la zone d'impact. Ce test nécessite une simple opération de distance entre le sommet et le centre du cercle, et la complexité devient $O(N_B)$.

4.2.4 Localisation du contact

L'algorithme précédent retourne deux ensembles de facettes E_A et E_B qui peuvent être potentiellement en collision. Localiser le contact consiste à déterminer les paires de facettes $(F_A^i, F_B^i) \in E_A \times E_B$ qui sont effectivement en collision. Pour que deux facettes (F_A^i, F_B^i) soient en collision, il faut qu'elles vérifient trois conditions:

Condition 1: $\vec{N_{F_A^i}} \times \vec{N_{F_A^i}} < 0$. où $\vec{N_{F_A^i}}$ est le vecteur directeur de la facette F_A^i dirigé vers l'extérieur de cette facette. C'est le cas (voir figure 4.1) des facettes (F_A^2, F_B^4) mais pas le cas de (F_A^1, F_B^4) pourtant les deux paires de facettes s'interpénètrent. Le contact géométrique entre F_A^1 et F_B^4 n'a pas de

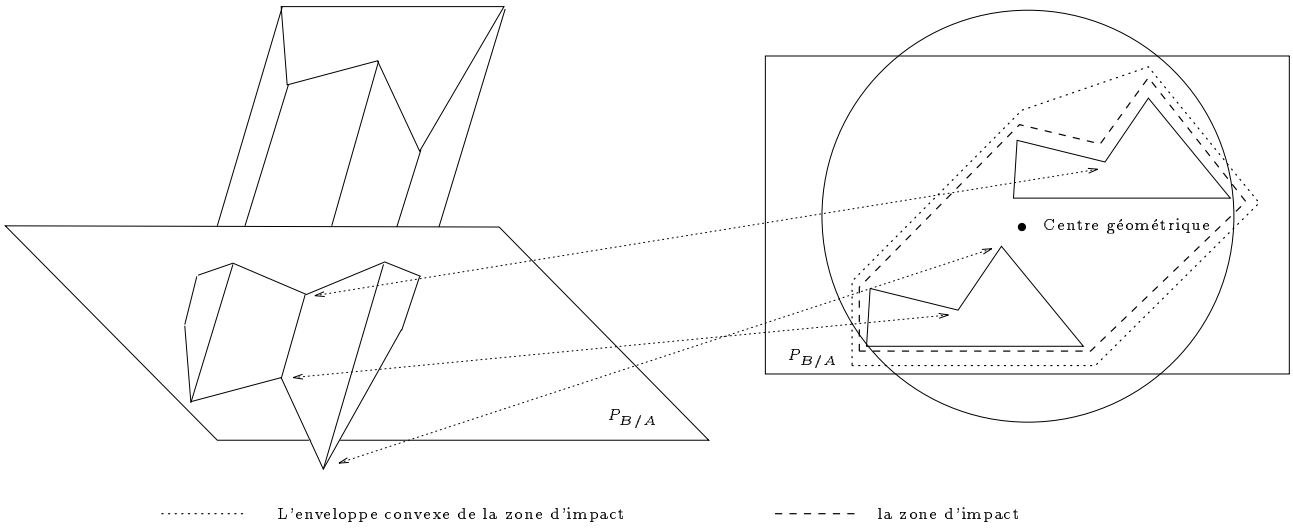


Figure 4.8: *Afin de réduire la complexité de l'algorithme, la zone d'impact doit être surestimée par un cercle.*

sens physique car entre ces deux facettes il y a la facette F_A^2 qui les empêche physiquement de se contacter.

Condition 2: *une partie de l'une se trouve derrière l'autre.* Deux facettes qui vérifient la condition 1 ne sont pas forcément en collision. C'est le cas de F_A^1 et F_B^6. Pour que les facettes soient en collision il faut qu'une partie de chacune d'elles se trouve derrière l'autre (car on utilise la méthode de la pénalité pour calculer la force de la collision. voir §4.3).

Condition 3: *la distance minimale entre elles est inférieure à un certain seuil.* Si les deux facettes s'interceptent alors la distance entre elles est nulle et elles sont en collision. C'est le cas de F_A^3 et F_B^2. Si les deux facettes ne n'interpénètrent pas alors il faut que la distance minimale entre elles soit inférieure à un certain seuil ϵ. Les deux paires de facettes (F_C^1, F_B^1) et (F_C^2, F_B^7) vérifient la condition 2 mais (F_C^2, F_B^7) ne sont pas en collision. Donc la condition 3 nous permet d'éliminer tel cas, car les interpénétrations tolérées par le système sont trop petites par rapport aux dimensions des objets (d'environ 10^{-7}).

Quand deux facettes sont en collision, la direction de la force de la collision est perpendiculaire sur l'une de ces deux facettes. En fait les deux facettes ne sont jamais parallèles. Il y a souvent un sommet de l'une qui percute l'autre. Dans ce cas la direction de la collision est la normale à l'autre facette. Quand les deux facettes sont parallèles, on peut choisir n'importe quelle normale.

Dans le cas du système *Robot*Φ, les polyèdres sont initialement connexes et les déformations sont le résultat d'une collision entre deux polyèdres. Dans ce cas, on peut faire l'hypothèse que tous les points de contact entre les deux polyèdres se trouvent du même côté, c'est-à-dire que A ne peut pas être en contact avec deux

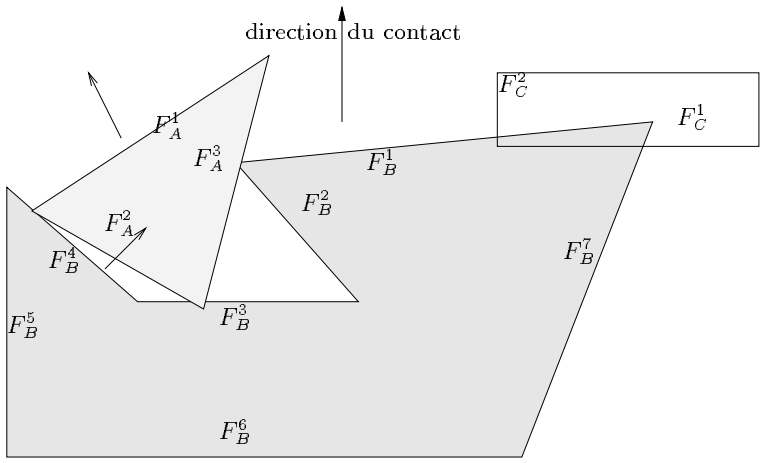


Figure 4.9: *Pour déterminer les facettes qui sont effectivement en collision il faut ajouter des testes supplémentaires.*

côtés opposés de B en même temps (figure 4.10). Cela nous permet d'ajouter un troisième critère d'élimination de facette qui est:

- Si $\vec{n}_{A/B} \times \vec{N_{F_A^i}} < 0$, où $\vec{N_{F_A^i}}$ est le vecteur directeur de la facette F_A^i dirigé vers l'extérieur de cette facette, alors la facette est invisible pour le polyèdre B et elle peut être éliminée. C'est le cas des facettes F_A^1, F_A^2, F_B^1, F_B^2, F_B^3 dans la figure 4.1.

De plus, on peut considérer dans ce cas que la direction de la force de la collision est constante quelle que soit la paire de facettes qui est en collision et que cette direction est égale à la direction du contact. Dans ce cas, on n'a pas besoin de la condition 3 qui peut être, dans le cas général, peu robuste. Le fait de considérer que les facettes F_C^2 et F_B^7 sont en collision n'est plus gênant dans ce cas, car la force de la collision résultante dans la direction du contact est nulle (voir §4.3).

4.3 Collision

Quand deux objets entrent en collision, ils se déforment localement (figure 4.11). Par conséquent, ils emmagasinent de l'énergie potentielle. Cette énergie se transforme en énergie cinétique et engendre la force de la collision, lorsque les deux objets retournent à leurs formes d'origine.

4.3.1 Calcul de la force de la collision

Trois méthodes principales permettent de calculer la force de la collision entre deux objets en mouvement (§1.4) :

La méthode de contrainte [4] qui consiste à résoudre analytiquement les contrainte de non-pénétration entre deux objets. Cette méthode exige que la trajectoire de

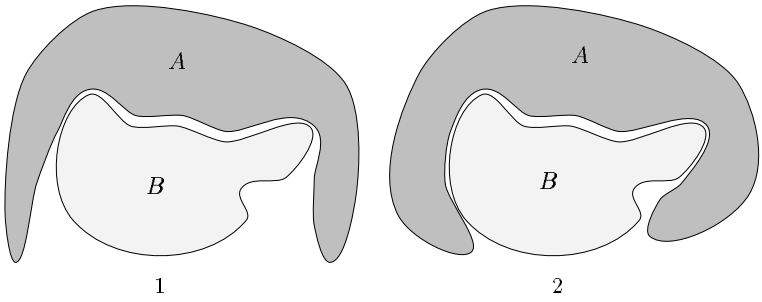


Figure 4.10: *Pour rendre notre algorithme plus robuste, on fait l'hypothèse qu'un polyèdre ne peut pas être en contact avec deux côtés opposés d'un autre. Alors, le cas* 1 *est accepté mais pas le cas* 2*. Cette hypothèse vient du fait que les polyèdres sont initialement convexes.*

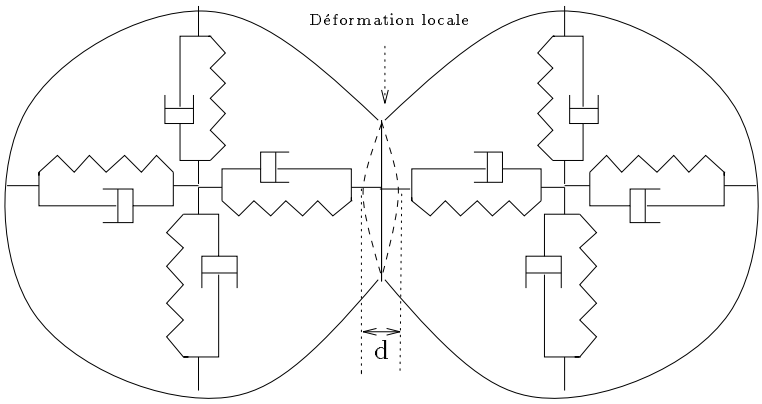


Figure 4.11: *La force de la collision est le résultat d'une déformation locale de la zone de contact. Sa valeur dépend du niveau de la déformation et de la matière.*

chaque objet soit complètement déterminée par les conditions initiales. Cela ne nous permet pas de contrôler le mouvement de l'objet. De plus la prise en compte de la force de frottement rend le calcul plus lourd [5, 8].

La méthode d'impulsion [80] consiste à calculer une force (dite impulsion) qui permet d'empêcher les deux objets de s'interpénétrer. cette méthode exige que la période de contact soit négligeable et que les deux objets soit parfaitement rigide. Ce qui n'est pas notre cas.

La méthode la plus adaptée (et en même temps la plus simple) pour calculer la force de collision entre deux objets, est la méthode de pénalité [81]. Le principe de ces méthodes consiste à générer dynamiquement une force répulsive qui dépend de la valeur de l'interpénétration fictive entre les deux objets. Le plus souvent la force répulsive est générée par un mécanisme de "ressort/amortisseur" (figure 4.11), et elle peut être donnée dans ce cas par la formule suivante:.

$$\vec{F_c} = \begin{cases} (-\lambda x - \mu \dot{x})\vec{k} & \text{si } x < 0 \\ \vec{0} & \text{sinon} \end{cases} \tag{4.1}$$

où λ est la rigidité de la collision, μ la viscosité de la collision (qui représente la dissipation de l'énergie), x est la distance entre les deux points qui sont en contact, $\dot{x}$ est la dérivé de x, et $\vec{k}$ est la direction de la collision (dirigé d'un point vers l'autre).

Cette méthode a cependant plusieurs inconvénients [73] §1.4. La force de la collision représente une discontinuité en 0, et le coefficient de restitution e[3] dépend de la masse de chaque objet [73]. Or le coefficient de restitution est une propriété intrinsèque de la matière (il ne doit donc pas dépendre de la masse). Il est prouvé qu'à faible vitesse d'impact et pour la plupart des matériaux dans le domaine élastique, le coefficient de la restitution peut être approché par [47, 41]:

$$e = 1 - \alpha v_i \tag{4.2}$$

Un modèle non linéaire a été proposé par [47] et étudié par [73] afin de surmonter ces problèmes. Ce modèle consiste à introduire une relation entre la force d'amortissement $-\mu\dot{x}$ et la distance d'interpénétration x. La force de la collision est alors donnée par:

$$\vec{F_c} = \begin{cases} (-\lambda x^n - \mu \dot{x} x^n)\vec{k} & \text{si } x < 0 \\ \vec{0} & \text{sinon} \end{cases} \tag{4.3}$$

où n est souvent très proche de 1 (elle dépend de la géométrie de la surface de la collision). L'avantage majeur de ce modèle est que le coefficient de restitution ne dépend que de la vitesse relative des deux objets. Si on choisit $\mu = \frac{3}{2}\alpha\lambda$, le coefficient de restitution, pour des faibles valeurs de α, vérifie l'équation 4.2[47]:

$$e = 1 - \alpha v_i$$

.

Donc ce modèle non linéaire va être intégré dans notre système pour calculer la force de la collision.

[3] e représente le rapport entre les vitesses relatives de deux points avant la collision v_i et leurs vitesses relatives après la collision v_0 : $v_0 = -e\, v_i$

4.3.2 Calcul de la contrainte de non-interpénétration

Quand les objets ne s'interpénètrent pas (c'est le cas quand les objets sont rigides et quand on utilise une méthode basée sur l'impact) on peut considérer que le contact est ponctuel, et on applique la force de la collision sur les deux points qui sont en contact [80, 68]. Quand les deux objets s'interpénètrent, l'hypothèse d'un contact ponctuel n'est plus valide, et dans ce cas il y a une infinité de points de contact. En pratique et pour des raisons évidentes de rapidité, on traite un nombre fini de ces points de contact en choisissant pour cela des points dont les fonctions sont significatives. Dans [89, 45] il est considéré que la force de la collision porte uniquement sur les sommets de chaque objet polyédrique ce qui ne permet pas de traiter les collisions de type arête/arête. Baraff [4] calcule la région de contact entre les deux facettes (figure 4.12) et applique la force de collision sur les sommets de cette région (la région est alors polygonale).

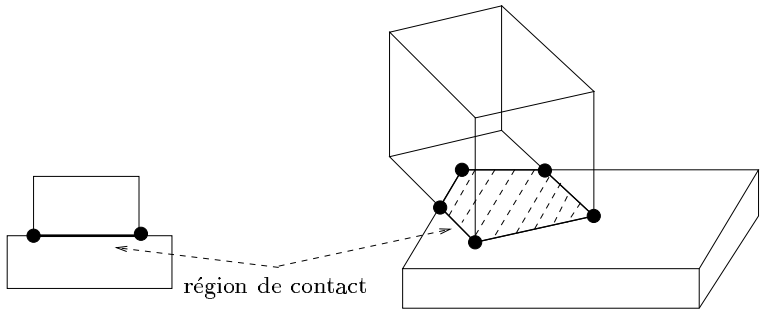


Figure 4.12: *Quand deux objets sont en contact permanent, les facettes sont dans le même plan et on peut facilement calculer la zone du contact [4].*

Dans notre cas, les objets peuvent se déformer. La déformation est approchée par l'interpénétration (figure 4.13) et les facettes ne sont donc pas forcément parallèles, et il y a simultanément plusieurs paires de facettes qui peuvent être en collision (cinq paires dans la figure 4.13). Pour chaque paire de facettes (F_i, F_j) on peut définir deux régions de contact, chacune liée à une facette. La région de contact liée à la facette i (resp. j) est l'intersection des deux facettes après les avoir projetées sur le plan de la facette i (resp. j) selon la direction du contact. Le plan d'une facette décompose l'espace en deux parties: l'extérieur du polyèdre et l'intérieur. Un objet qui se trouve dans la partie extérieure est dit devant la facette, sinon il est derrière. Seulement la partie de la facette F_i (resp. F_j) qui se trouve derrière la facette F_j (resp. F_i) est réellement en collision. Pour cela avant de calculer la région du contact, il faut éliminer la partie de chaque facette qui ne peut pas être en collision (figure 4.13). Pour être plus réaliste on introduit la notion du volume d'interpénétration entre chaque paire de facette. Ce volume est par définition le volume compris entre deux régions de contact. La force de la collision entre chaque paire de facettes dépend de ce volume.

Pour calculer la force de la collision entre deux polyèdres, on calcule la force de la collision entre chaque paire de facettes $\{F_A^i, F_B^j\}$ (figure 4.14.a) retourné par

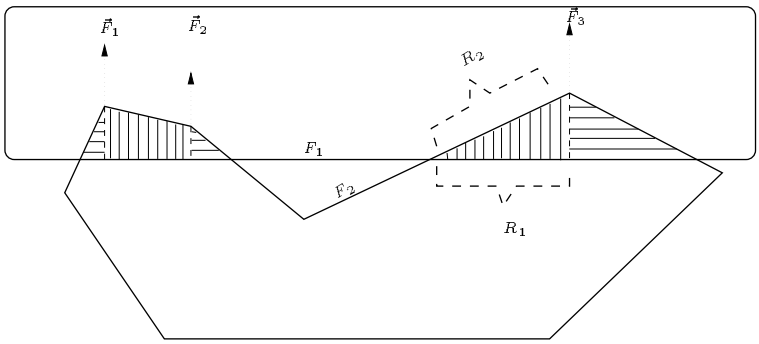


Figure 4.13: *Pour chaque paire de facettes (F_i, F_j) on définit deux régions de contact R_i et R_j qui sont obtenues par projection sur chaque facette après avoir éliminé les parties de chaque facette qui ne peuvent pas être en collision. Les parties hachurées sont les différents volumes d'interpénétrations.*

l'algorithme de la localisation du contact (§4.2). Pour calculer la force de la collision entre deux facettes, on procède comme suit:

1. On élimine la partie de F_A^i qui se trouve derrière F_B^j et la partie de F_B^j qui se trouve derrière F_A^i. On obtient deux nouvelles facettes polygonales (P_B et P_A) qui sont effectivement en contact (figure 4.14.b).

2. Pour trouver la région de contact R_A (resp. R_B), on calcule l'intersection de P_B et P_A après les avoir projetées sur le plan de F_A^i (resp. F_B^j) selon la direction du contact. Ces régions de contact ont le même nombre de sommets. Les lignes liant deux sommets correspondants (R_A^k, R_B^k) de ces deux régions de contact sont parallèles. Dans la figure 4.14 les deux régions ont deux sommets en commun ($R_A^3 = R_B^3$ et $R_A^4 = R_B^4$).

3. Le volume d'interpénétration v_i^j entre les deux facettes est par définition le volume compris entre les deux régions de contact. La force de la collision dépend de ce volume et on la distribue entre les sommets de chaque région de contact. Pour chaque paire de sommets R_A^k et R_B^k on calcule la contribution de ces deux sommets au volume d'interpénétration v_k. Cette contribution est proportionnelle à la distance les séparant :

$$v_k = \frac{x_k}{\sum_n x_n} v_i^j$$

.

où x_k est la distance entre les deux sommets R_A^k et R_B^k. La force de la collision agissant sur chaque paire de sommets est alors donnée par:

$$F_k = (-\lambda v_k - \frac{3}{2}\alpha\lambda \dot{x}_k v_k)$$

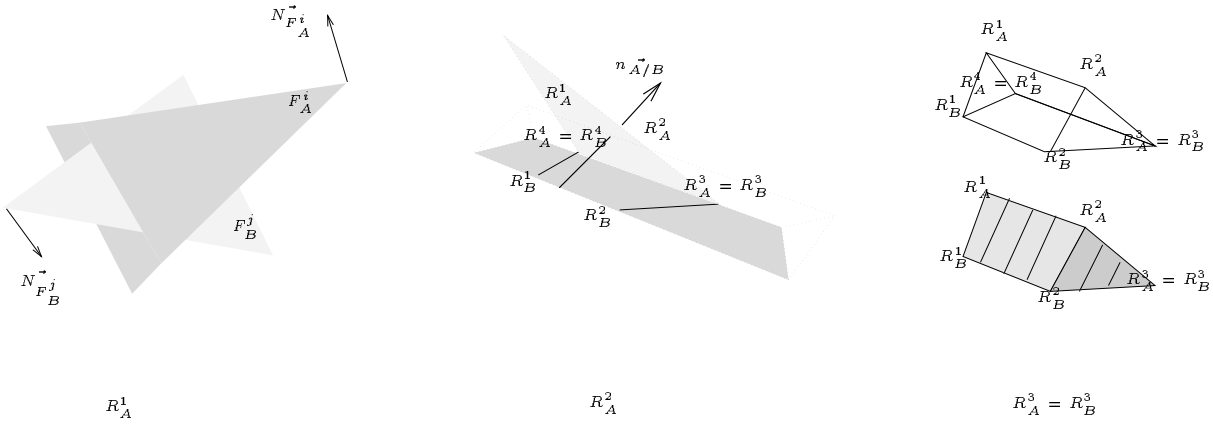


Figure 4.14: *(a) Deux facettes F_A^i et F_B^j sont en contact. (b) Les régions de contact sont $R_A^1, R_A^2, R_A^3, R_A^4$ et $R_B^1, R_B^2, R_B^3, R_B^4$. (c) $R_A^1, R_A^2, R_A^3, R_A^4, R_B^3, R_B^4$ est le volume compris entre F_A^i et F_B^j.*

Seules les particules ont des masses. Les facettes triangulaires (si elles ne sont pas triangulaires, on ne peut pas assurer qu'elles restent coplanaires pendant les déformations de l'objet) sont définies implicitement par trois particules et elles ne possèdent aucune masse et elles ne peuvent pas réagir. Donc, la force $\vec{F}$ qui agit sur un point d'une facette, doit être distribuée entre les trois particules (P_1, P_2, P_3) qui définissent cette facette. Cette distribution se fait en respectant l'amplitude de la force et de son moment:

$$\begin{array}{ll} \vec{F_c} = \vec{F_1} + \vec{F_2} + \vec{F_3} & amplitude \\ \vec{F_c}\vec{P_c} = \vec{F_1}\vec{P_1} + \vec{F_2}\vec{P_2} + \vec{F_3}\vec{P_3} & moment \end{array} \tag{4.4}$$

Comme la direction de la force est déjà connue, le système 4.4 a une et une seule solution.

4.4 Le frottement

Quand deux objets en mouvement sont en contact, ils subissent une force de frottement. Le frottement est une notion statistique. Elle est la résultante d'un grand nombre de micro-collisions au niveau de la région du contact (figure 4.15).

Pour modéliser ces micro-collisions il faudrait avoir une représentation très fine des surfaces des objets. Cela n'est évidement pas possible.

En pratique, la valeur de la force de frottement $|\vec{F_f}|$ peut être considérée comme proportionnelle à la valeur de la force normale $|\vec{F_n}|$ appliquée par l'un des deux objets sur l'autre (loi de Coulomb). Donc, cette force peut être définie par la relation

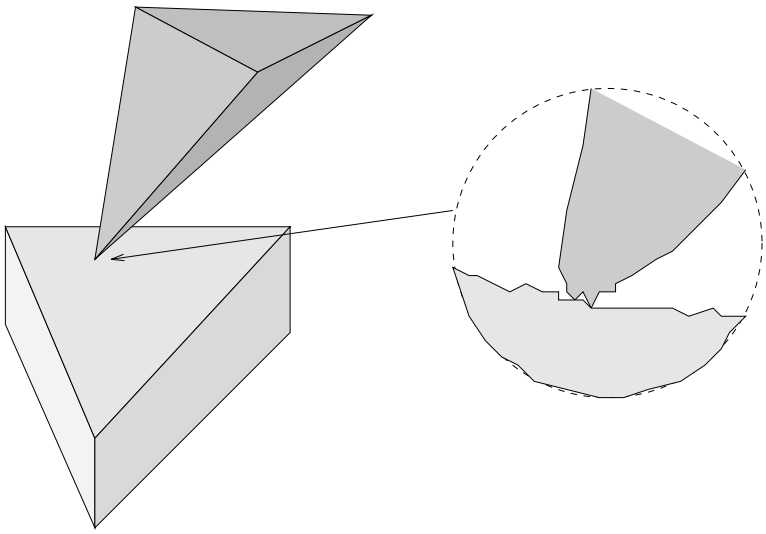


Figure 4.15: *La force du frottement est la résultante d'un grand nombre de micro-collisions au niveau de la région du contact*

suivante :

$$\vec{F_f} = \begin{cases} -c\,|\vec{F_n}|\frac{\vec{V}}{|\vec{V}|} & \text{Si } \vec{V} \neq \vec{0} \quad (glissement) \\ -s\,|\vec{F_n}|\frac{\vec{F_t}}{|\vec{F_t}|} & \text{Si } |\vec{F_t}| > s|\,\vec{F_n}| \quad (\text{décollage}) \\ -\,\vec{F_t} & \text{Sinon} \quad (\text{adhérence}) \end{cases} \tag{4.5}$$

où c est le facteur cinétique du frottement, s le facteur statique du frottement ($s > c$), $\vec{V}$ la vitesse relative de deux points du contact, et $\vec{F_t}$ la force tangentielle appliquée par l'un de deux points sur l'autre.

Les points sur lesquelles la force du frottement est appliquée sont toujours les points correspondants aux régions de contacts. Pour appliquer la formule 4.5, il faut calculer la force normale appliquée par chaque point sur l'autre, et la vitesse relative de ces deux points. La force appliquée par un point sur son correspondant est égale à leur contribution à la force de la collision. Leur vitesse relative est la différence entre leurs vitesses absolues. Le problème réside dans le fait que les sommets d'une région de contact ne sont pas forcément des particules, il faut donc pouvoir calculer la vitesse d'un point quelconque qui appartient à une facette triangulaire déformable où les vitesses absolues de ces sommets sont connues.

Si p est un point d'une facette triangulaire abc alors:

$$\vec{P_p} = \alpha\vec{P_a} + \beta\vec{P_b} + \gamma\vec{P_c}$$

Comme les sommets d'une facette triangulaire abc se trouve toujours dans le même plan, alors la seule manière de se déformer est de changer la distance relative entre les différents sommets. Cette déformation est une transformation projective qui conserve le rapport, donc α, β, et γ sont constantes. En dérivant on obtient:

$$\vec{V_p} = \alpha\vec{V_a} + \beta\vec{V_b} + \gamma\vec{V_c}$$

4.5 La Viscosité

Quand un objet est en mouvement, il interagit avec l'environnement (air, eau). Cette interaction est le résultat d'un grand nombre de micro-collisions entre l'objet et les molécules qui constituent l'environnement. Ces micro-collisions engendrent une force qui peut changer le comportement de l'objet. Par exemple, un objet en chute libre tourne pendant sa chute à cause de cette force de viscosité de l'air (figure 4.16).

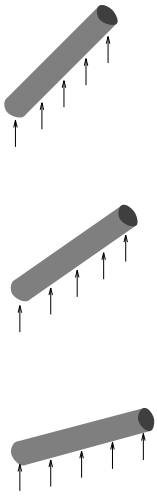


Figure 4.16: *La force de la viscosité est la cause de la rotation d'un objet en chut libre.*

La force engendrée par la viscosité du milieu dépend de la nature de la surface de l'objet, de sa vitesse et de sa forme géométrique (elle peut être négligée quand l'objet se déplace lentement dans un environnement peu visqueux).

La prise en compte de cette force de viscosité, nécessite d'associer une représentation dynamique de l'environnement qui peut être liquide ou gazeuse. Cette représentation serait très coûteuse en espace et en temps, car elle nécessiterait une discrétisation très fine de l'environnement en particules. En pratique, on utilise une représentation macroscopique de la force résultante de ce phénomène, puisque sa valeur en un point de l'objet $\vec{F_v}$ est proportionnelle à la vitesse de ce point[4]: $\vec{F_v} = -k\vec{V}$, ou k est le facteur de la viscosité et $\vec{V}$ la vitesse du point en question. Pour une surface infinitésimale δs, on a $\vec{F_v^s} = -k\vec{V_s}$. Pour une surface S, il faut calculer la somme des forces de viscosité en chaque point de cette surface, ce qui donne l'expression suivante:

$$\vec{F_v} = -k \int_S \vec{V_s} \delta s = -k \int_S \frac{\delta \vec{P_s}}{\delta t} \delta s = -k \frac{\delta}{\delta t} \int_S \vec{P_s} \delta s$$

$$\vec{F_v} = -k\, S \frac{\delta \vec{P}_{I_C}}{\delta t} = -k\, S\, \vec{V}_{I_C}$$

[4]Cette approximation est valide pour un objet qui se déplace lentement dans l'air, ce qui est le cas des robots que nous considérons. Dans un liquide, cette force est proportionnelle au carré de la vitesse de chaque point, et elle a la forme $\vec{F_v} = -k\,\vec{V}^2$

où, Ic dénote le centre d'inertie de la facette. Evidemment, la force de la viscosité ne dépend pas seulement de la surface d'une facette, mais elle dépend aussi de l'orientation de cette facette par rapport à la direction du mouvement. La force est maximale quand le vecteur directeur de la facette a la même direction que la vitesse, c'est à dire $\vec{V} \times \vec{n} = |\vec{V}|$. Elle est nulle quand la facette est cachée par d'autres facettes dans la direction du mouvement (soit lorsque $\vec{V} \times \vec{n} < 0$, où $\vec{n}$ est le vecteur normal externe de la facette). Par conséquent, la force $\vec{F}_v$ peut être définie par la relation suivante:

$$\vec{F}_v = \begin{cases} -k\, S\, (\vec{V}_{Ic} \times \vec{n}).\frac{\vec{V}}{|\vec{V}|} & \text{si } \vec{V}_{Ic} \times \vec{n} > 0 \\ 0 & \text{sinon} \end{cases} \tag{4.6}$$

Dans notre cas, les facettes sont triangulaires. Chaque facette a trois sommets (P_1, P_2, P_3), et la vitesse du centre d'inertie I_c est la vitesse moyenne des trois sommets:

$$\vec{V}_{Ic} = \frac{1}{3} \sum_{i=1}^{3} \vec{V}_i$$

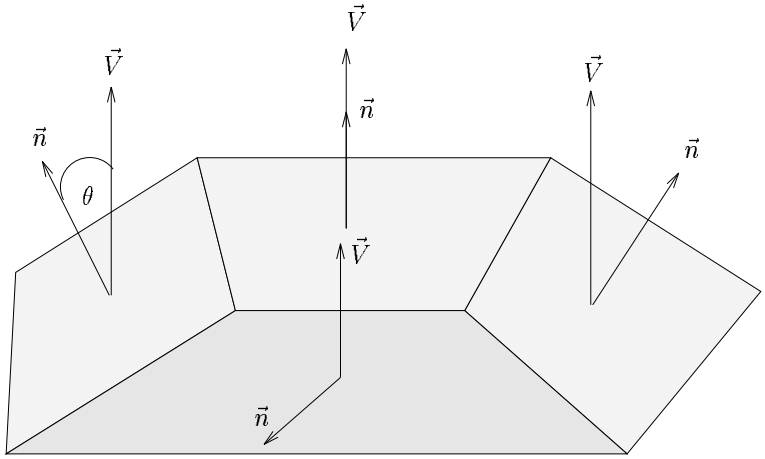


Figure 4.17: *La force de la viscosité, appliquée sur une facette, dépend de la projection de la surface de la facette sur un plan, dont le vecteur directeur est colinéaire avec la vitesse de cette facette.*

4.6 Expérimentations

La direction du contact entre deux polyèdres a été observée pendant plusieurs tests. Le nombre moyen d'itérations nécessaire pour trouver la direction du contact était très proche de 1 ($I \simeq 1$).

La figure 4.18 montre un contact entre une sphère rigide et un objet flexible. On peut remarquer que la divergence locale de notre algorithme de détection de contact

fait que la direction du contact reste constante pendant la collision, pourtant c'est plus facile de séparer les deux objets par une translation dans le sens inverse de la direction de contact.

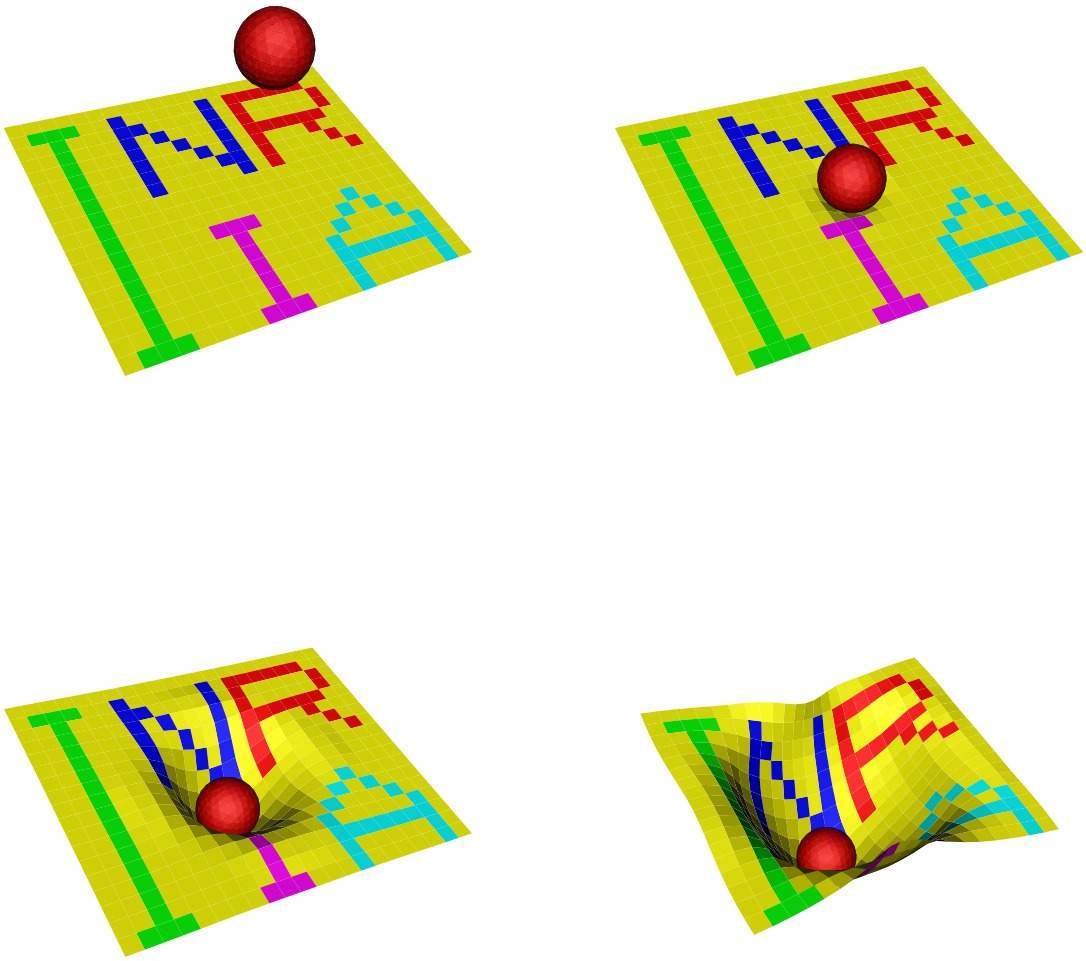


Figure 4.18: *L'algorithme de la détection de contact est robuste pour des valeurs d'interpénétration significatives entre les deux enveloppes convexes de deux polyèdres.*

La force de la collision engendrée par ce contact fait que les deux objets restent toujours séparés.

La figure 4.19 montre l'évolution de la direction du contact, quand un polyèdre concave (la base du polyèdre est concave) chute sur une table. Cette simulation dynamique prend 34 secondes pour être accomplie (2000 itérations). La moitié de ce temps est consacré à la détection du contact, presque 0.0085 seconde par détection de contact.

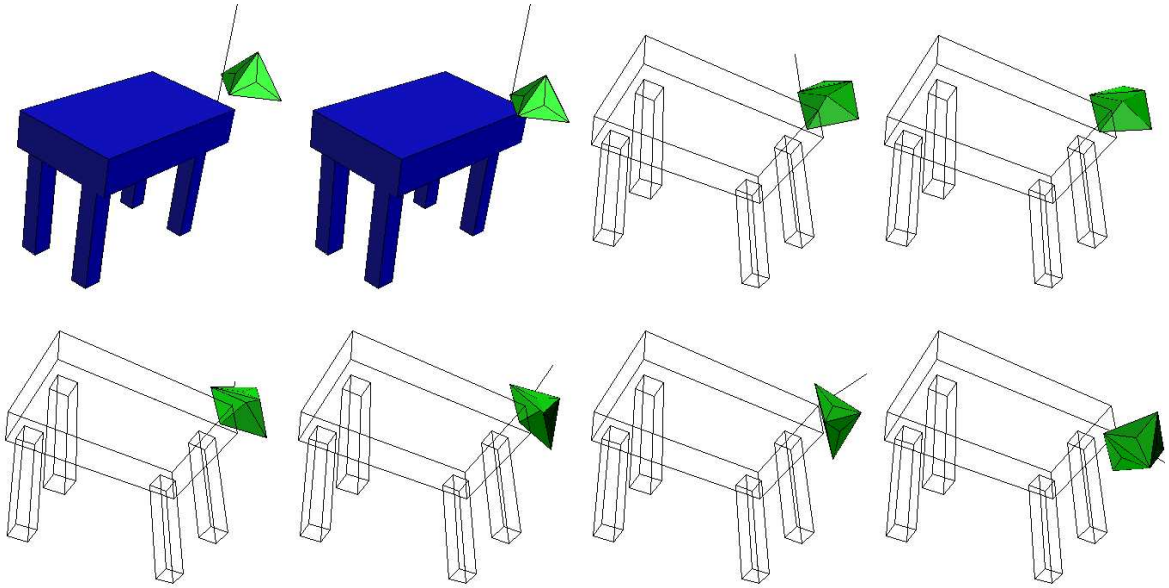


Figure 4.19: *L'évolution de $\vec{n}_{A/B}$ pendant la collision. La direction du contact est représentée par une ligne noire. Au début cette direction était perpendiculaire à la surface de la table. A cause de la force de la collision et à la force de la pesanteur, la pyramide bascule sur le sommet et la direction de contact passera par ce sommet.*

La figure 4.20 montre aussi l'effet de la force de la collision et de la force de frottement sur le mouvement des objets. Elle montre un cylindre qui roule sur le sol. La force de la collision garde le cylindre sur le sol et la force du frottement fait que le cylindre roule au lieu de glisser. En fait, la partie du cylindre qui est en contact avec la piste est accrochée avec elle par la force de frottement, tandis que le reste est en mouvement libre, ce qui cause la rotation. Pour une force de frottement faible, le cylindre tourne et glisse en même temps. Pour une force de frottement nulle, le cylindre glisse sans tourner. Le facteur statique du frottement contrôle le rapport glissement/rotation et le facteur cinétique du frottement contrôle la vitesse de translation du cylindre par rapport à la piste dans le cas d'un glissement. Le cube, par contre, bascule sous l'effet du frottement. Comme certaines parties du cube arrivent sur le sol avec une vitesse initiale très importante, la force de la collision est aussi importante et le cube rebondit sur le sol plusieurs fois. Le pas de temps moyen dans cet exemple est d'environ 0.005 seconde.

4.7 Conclusion

On a présenté dans ce chapitre un algorithme incrémental, qui permet de détecter et localiser le contact entre deux polyèdres déformables. Cet algorithme donne à la sortie un ensemble de paires de facettes qui sont effectivement en contact, et fournit la distance entre les enveloppes convexes de ces polyèdres. Cet algorithme comporte deux phases. La première phase consiste à trouver une boîte englobante de la zone d'interpénétration entre les deux enveloppes convexes et à éliminer toutes les facettes

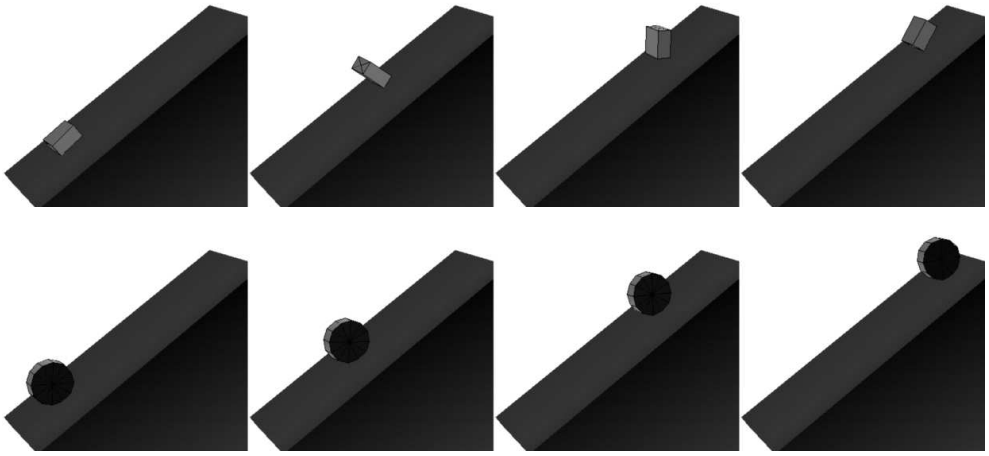


Figure 4.20: *La différence entre le comportement d'une roue qui tourne sur la terre tout en restant collée avec elle, et le comportement d'un cube qui quitte la terre plusieurs fois pendant la chute.*

qui n'appartiennent pas à cette boîte. La complexité de cette phase est de $C = k.n$ (où n est le nombre de facettes et k le nombre d'itérations), le coefficient k dépend de la valeur initiale de la direction du contact. En pratique la valeur moyenne de k est trop petite et très proche de 1. Donc, l'algorithme converge en temps linéaire. La deuxième phase consiste à trouver les facettes qui sont effectivement en collision. Cela nécessite $n_1 \times n_2$ opérations où n_1 (resp. n_2) est le nombre de facettes du premier (resp. deuxième) polyèdre qui n'ont pas étaient éliminés. Ce nombre est souvent très faible.

La distance entre les deux polyèdres doit être initialement positive pour initialiser correctement l'algorithme. L'utilisation de cette méthode pour détecter le contact ne peut pas rendre le système optimal, car on traite tous les objets de la même façon sans faire de différence entre un objet rigide et un objet déformable. On peut optimiser davantage le système si on fait cette différence et si on intègre d'autres approches [69, 80, 33] avec la nôtre.

On a présenté également les trois types d'interaction prises en compte par notre modèle: collision, frottement et viscosité du milieu. Pour calculer la force de la collision on a adopté une méthode de pénalité non linéaire, où la force de la collision est représentée par un ressort et un amortisseur non-linéaires [47] liés en parallèle. Ce modèle non-linéaire permet d'avoir un coefficient de restitution qui ne dépend que de la vitesse relative des deux objets ce qui est la réalité physique du phénomène. La force du frottement est représentée par la loi de Coulomb, où on fait la distinction entre le frottement statique et le frottement cinétique. La force de la viscosité est représentée par une force proportionnelle à la vitesse de l'objet et elle dépend de la surface de l'objet qui est pratiquement en contact avec l'environnement dans la direction du mouvement.

Les forces d'interaction sont appliquées sur un nombre limité de points, qui sont les sommets des régions effectives du contact . On peut être davantage réaliste si on

applique ces forces sur tous les points des régions de contact. Cela nécessite de faire une intégrale numérique que l'on ne sait pas calculer, pour le moment, que d'une manière numérique. Les méthodes numériques consistent à discrétiser la surface pour calculer l'intégrale. Cela n'est pas suffisant et il faut donc trouver un moyen pour calculer ces intégrales d'une manière analytique.

Chapter 5

La génération du mouvement : Applications robotiques, médicales et graphiques

5.1 Introduction

La simulation dynamique définit un *monde virtuel* dans lequel le comportement des objets et des robots est cohérent avec le monde réel et leurs mouvement peut être contrôlés par les contrôleurs classiques. Cette possibilité est très utile pour planifier des tâches robotiques qui introduisent des interactions complexes entre le robot et son environnement. Ces interactions ne pouvaient pas être prises en compte par un modèle purement géométrique. Le monde virtuel permet de valider les contrôleurs du mouvement avant de les appliquer sur un robot réel. La validation dans un monde virtuel permet d'éviter les dégâts de matériels qui peuvent se produire à cause d'une action erronée. De plus, on peut envisager des algorithmes automatiques (voir §6) pour régler les paramètres d'un contrôleur.

Le mouvement d'un objet dans le monde réel est généré par l'intermédiaire d'un moteur ou d'un actionneur. Dans le monde virtuel, nous avons de plus la possibilité de générer le mouvement par une "force virtuelle". Une force est dite virtuelle si elle n'est pas engendrée par un actionneur. Une telle force peut être appliquée à n'importe quel endroit de l'objet contrôlé.

La force virtuelle sert, évidemment, à reproduire des mouvements engendrés par des actionneurs très complexes tels que le mouvement passif du ligament de genou (§5.5) qui est engendré lors d'une manipulation par un opérateur humain.

De plus le simulateur dynamique permet à ces forces de se propager à l'intérieur de l'objet jusqu'aux articulations et actionneurs et de les faire bouger en fonction du mouvement voulu. Cela permet d'engendrer automatiquement la cinématique inverse d'une chaîne articulée ou d'un véhicule sous la condition de non-glissement.

Les forces virtuelles peuvent être engendrées par une loi de commande quelconque ou par l'intermédiaire de "contraintes géométriques" comme c'est le cas pour la force de la collision.

Dans ce chapitre nous expliquons, comment les forces virtuelles peuvent servir à générer[1] le mouvement d'un objet modèle, et quel est le lien entre ces forces et le monde réel. Ensuite nous présentons trois applications: une application robotique concernant la saisie par une main articulée, une application médicale concernant la simulation du comportement du ligament du genou pendant le mouvement passif et une application graphique concernant la génération des effets spéciaux pour les appliquer sur des images $2D$ qui évolue dans le $3D$.

5.2 Le génération du mouvement par une force

Une propriété principale de la simulation dynamique, est la possibilité de manipuler un objet en agissant seulement par une force externe. Cette force se propage à l'intérieur de l'objet et elle cause les mouvements des différentes parties. Si le mouvement de cet objet le fait interagir avec d'autres objets dans l'environnement, l'effet de cette interaction sera également simulé. On distingue deux cas : le cas d'un objet rigide, et celui d'un objet, déformable ou rigide, articulé.

5.2.1 Objet rigide

Une méthode directe pour amener un objet rigide (par exemple un robot mobile à deux degrés de liberté sans contraintes non holonomes) à une nouvelle position voulue, est de le "tracter" par une force vers sa nouvelle position. La position finale (le but) peut être définie par une "duplication virtuelle" de l'objet qui se trouve déjà à cette position. L'objet est lié à sa "duplication virtuelle" à l'aide des connecteurs de force où la force engendrée est proportionnelle à la distance et à la vitesse relative de l'objet par rapport à son but (un contrôleur PD[2]). Le contrôleur PD se comporte comme un ressort/amortisseur (figure 5.1) qui lient l'objet avec sa duplication virtuelle. Le choix de la position et du type de connecteur dépend de la tâche à accomplir. Si la position finale est complètement définie (position et orientation), il faut relier par des connecteurs linéaires (LS) au moins deux particules[3] de l'objet avec les particules correspondantes de sa "duplication virtuelles". Les connecteurs TS peuvent être utilisés pour ajouter des contraintes sur la trajectoire (un morceau d'un arc du cercle, voir figure 5.1)

5.2.2 Objet articulé rigide ou déformable

Pour amener l'outil terminal d'une chaîne articulée à une position voulue, il faut d'abord rendre ses articulations passives(5.2). Une articulation est passive si elle n'exerce aucune force active sur le robot, autrement dit sa rigidité λ est nulle. Par contre la viscosité μ de cette articulation a une valeur relativement grande, qui

[1] Nous n'abordons pas dans ce chapitre le problème du contrôle dans le sens "automatique" du terme, mais nous utilisons des méthodes paramétrées pour générer le mouvement d'un objet et nous proposons dans le chapitre suivant de régler ces paramètres en utilisant les algorithmes génétiques

[2] Un contrôleur PID (proportionnel, intégral et dérivé), ou d'autres types de contrôleurs peuvent être également utilisés.

[3] Dans le cas où on traite un objet capable de se déplacer en $3D$, il faut au moins 4 particules non coplanaires

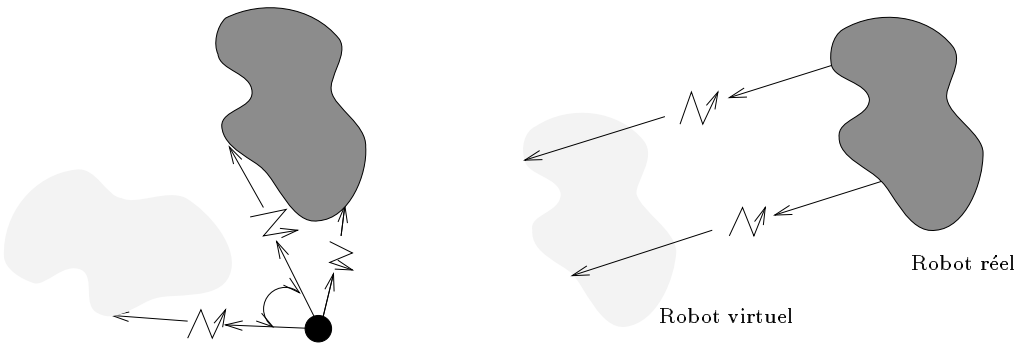


Figure 5.1: *Le génération du mouvement d'un objet rigide.*

permet de simuler le frottement cinétique au niveau de cette articulation (pas de frottement statique au niveau des articulations). Quand on applique un moment sur une articulation passive, elle bouge mais elle s'arrête dès que la force disparaît. Si on applique une force sur l'outil terminal de cette chaîne articulée, elle bouge et ses articulations passives se configurent automatiquement pour suivre ce mouvement. La direction du mouvement sera celle de la force (à condition que la vitesse initiale soit nulle), et ceci nous permettra de mettre l'outil terminal à l'emplacement désiré. La force à appliquer sur l'outil terminal peut être engendrée par n'importe quelle loi de commande (PD ou PID, etc) et nous avons la possibilité de régler d'une manière expérimentale et automatique les paramètres de cette loi(§6).

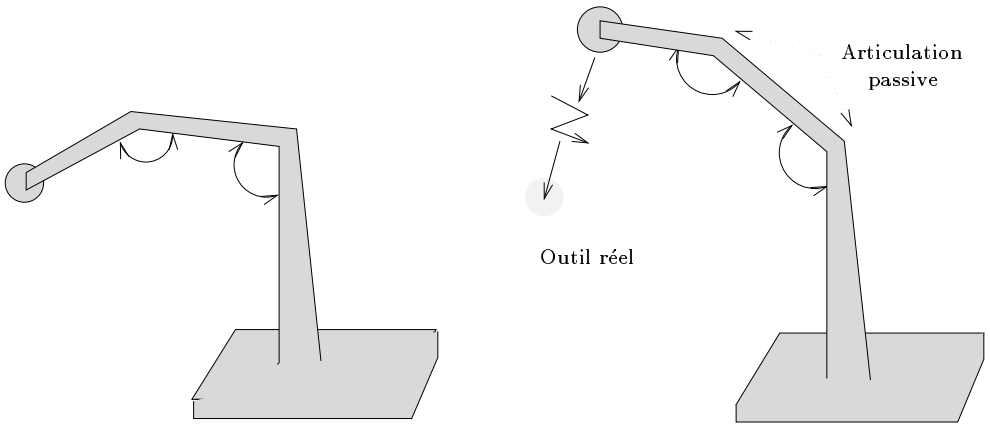


Figure 5.2: *Le génération du mouvement d'une chaîne articulée.*

Bien que la force virtuelle soit appliquée sur l'outil terminal de la chaîne articulée et que la trajectoire à suivre soit donnée dans l'espace du travail, le résultat du mouvement du robot est la trajectoire dans les deux espaces : celui du travail et celui de la configuration q_t (voir figure 5.4) et c'est q_t qu'il faut donner au robot réel pour qu'il exécute le même mouvement. Cette trajectoire est donnée en fonction du temps ce qui nous permet aussi de trouver la vitesse $\dot{q}_t$ et les accélérations $\ddot{q}_t$

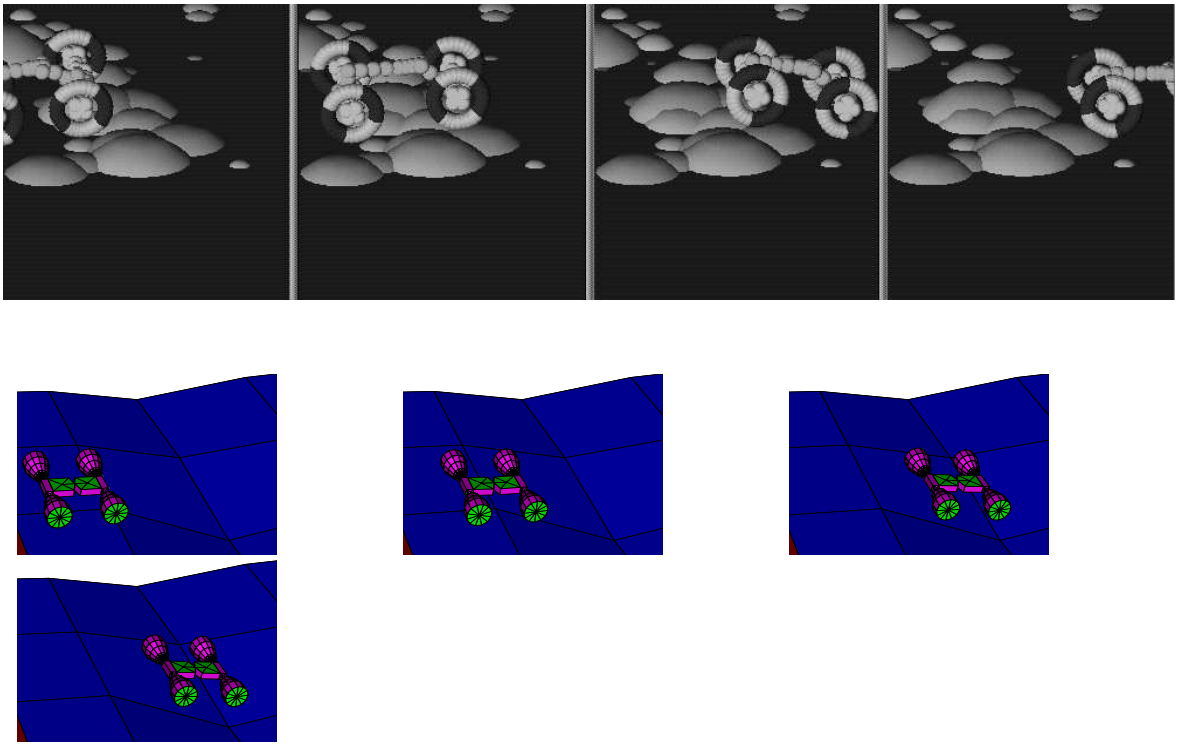


Figure 5.3: *Le mouvement de ce véhicule est causé par une force attractive dans la direction voulue et le véhicule se configure automatiquement pour s'adapter à la forme du sol.*

des articulations. Nous n'avons donc pas besoin calculer la cinématique inverse de la chaîne. Si les contraintes gémétriques (dues soit des articulations du robot soit de l'environnement) empêche le robot d'atteindre le but final, le robot modèle sera bloqué dans le minimum local correspondant.

La simulation dynamique nous permet aussi de calculer la cinématique directe : en appliquant un couple quelconque sur les articulations, l'organe terminal se positionne automatiquement à la position qui correspond à la configuration actuelle. Pour appliquer un couple sur une articulation, il suffit de changer sa position d'équilibre θ_0 (§2.2.3).

Un objet déformable (le cas du ligament du genou §5.5) peut être vu comme un objet articulé avec un nombre infini d'articulations. En pratique, chaque particule est le centre d'une articulation parfaite.

La figure 5.3 montre l'évolution de deux véhicules sur le sol. Les forces virtuelles sont appliquées sur le bout du châssis et elles tendent à l'obliger à suivre une trajectoire donnée. Les différentes parties du véhicule se configurent automatiquement en fonction de la force du contrôle et des forces d'interaction avec le sol. Le véhicule du haut a un châssis déformable pour s'adapter mieux à la forme du terrain. Le véhicule du bas est représenté par 500 particules. Il y a une articulation au milieu du châssis et les roues sont articulées avec le châssis. Le but est de valider la possibilité d'exécuter une trajectoire géométrique donnée [22, 67].

La figure 5.4 montre une manipulation d'un cylindre par un robot. Le robot est représenté par 160 particules, 3 articulations horizontales et une articulation verticale. Le cylindre par 36 particules qui forment un objet parfaitement rigide. Le robot est obligé à bouger dans la direction du cylindre par une force de contrôle de type *PD* dirigée vers le cylindre et appliquée sur l'outil terminal du robot. Les articulations du robot sont passives et elles suivent automatiquement le mouvement de l'outil terminal du robot. Dans cet exemple deux interactions sont considérées : l'interaction entre le robot et le cylindre, et l'interaction entre le cylindre et le sol. Le cylindre est initialement dans une position d'équilibre. La collision entre le robot et le cylindre fait sortir le cylindre de cet équilibre et le faire glisser sur la piste inclinée. Le pas de temps moyen pour réaliser cet exemple est de 0.0003 seconde.

5.3 Le contrôle par des contraintes géométriques

Un autre avantage de la simulation dynamique est la possibilité d'interaction entre les différents objets dans la scène. Cette possibilité peut être également utilisée pour contrôler le mouvement d'un robot en imposant certaines contraintes géométriques. Ces contraintes géométriques peuvent représenter un obstacle ou un objet à manipuler.

La force de la collision (§4.3) avec un obstacle tend à annuler la composante normale de la vitesse de l'objet mobile et elle l'oblige à suivre la frontière de l'obstacle. Quand l'obstacle est convexe, cela peut être suffisant pour l'éviter. Sinon des obstacles virtuels peuvent être ajoutés pour rendre l'obstacle réel convexe et pour "boucher" les minima locaux[4]. La figure 5.3 montre un véhicule qui se reconfigure automatiquement pour s'adapter à la forme du sol. Cette reconfiguration est due à la force de la collision avec le sol.

La figure 5.6 montre comment le contrôle par des contraintes géométriques aide à simuler l'assemblage passif lorsque l'on cherche à assembler deux objets dont les formes géométriques sont cohérentes.

Les contraintes géométriques aident également à simuler le buté mécanique des articulations du robot.

La saisie d'un objet par une main articulée est aussi un autre exemple de contrôle par contraintes géométriques (§5.4).

5.4 La saisie par une main articulée

Ce travail a été réalisé en coopération avec Christian Bard (équipe SHARP) dans le contexte de la manipulation automatique, et pendant sa thèse qu'il l'a soutenu en 1995 [10].

La stabilité de la prise d'un objet par une main articulée dépend des paramètres physiques tels que la nature de la force de la collision entre la main et l'objet (collision élastique ou plastique), la force de frottement, et aussi de la configuration géométrique de la main qui fait que l'objet peut rester en contact avec la main.

[4] Un minimum local es rencontré quand l'objet représente une concavité au point de collision.

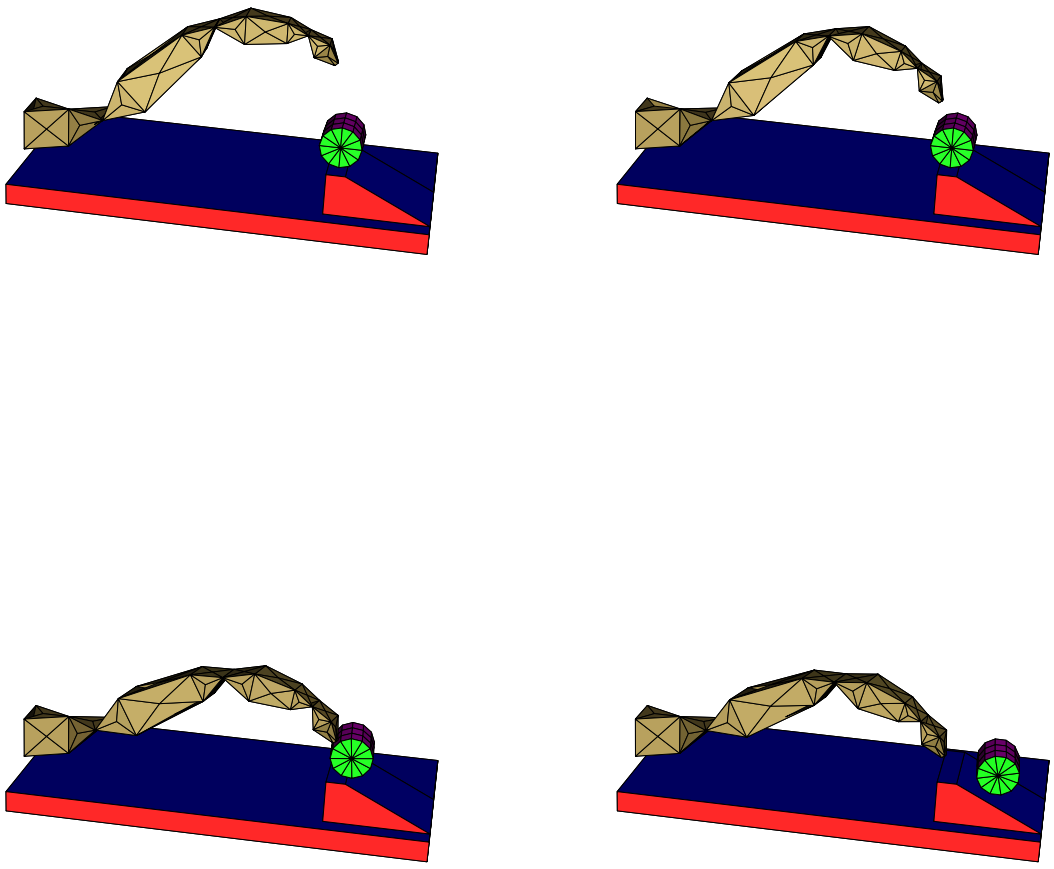


Figure 5.4: *La manipulation d'un cylindre par un robot où apparaît en même temps l'effet de la force de contrôle qui est appliquée sur l'outil terminal du robot, la force de la collision, et la force de frottement. Les articulations se configurent automatiquement pour suivre l'outil terminal du robot. Le pas de temps moyen est d'environ* 0.0003 *seconde.*

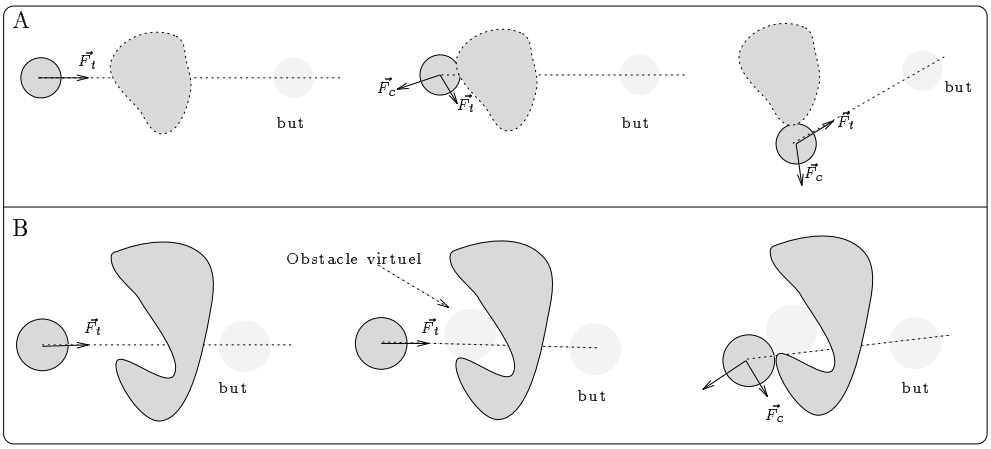


Figure 5.5: *L'évitement d'obstacle en utilisant de contraintes géométriques.*

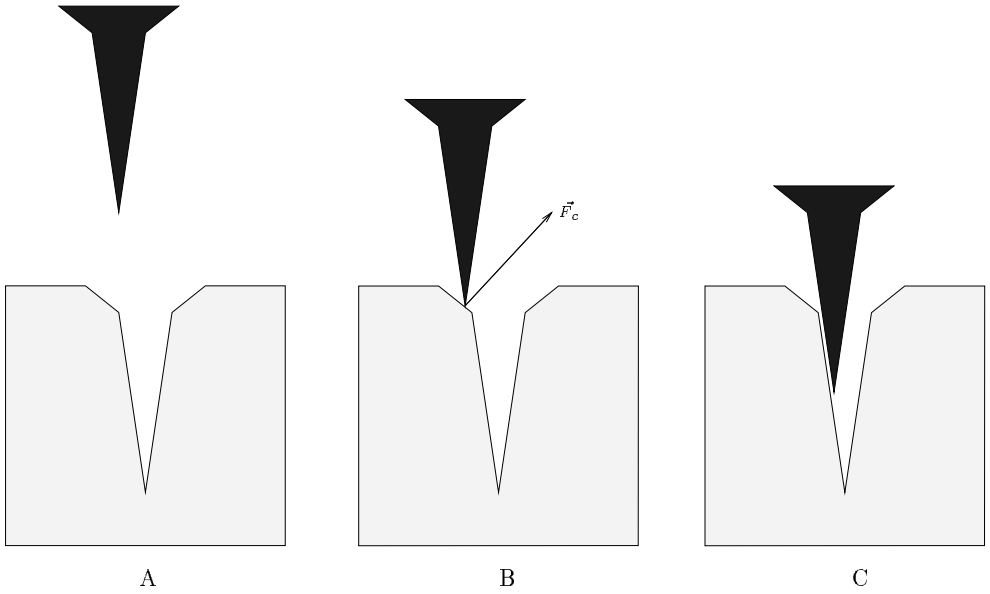


Figure 5.6: *La force de la collision permet de guider automatiquement l'opération de l'assemblage.*

Etant donnée une stratégie de saisie (chemin d'accès + pré-configuration[5] géométrique, voir figure 5.7), le simulateur dynamique sert à vérifier si cette stratégie est ou pas dynamiquement stable [10, 57, 56].

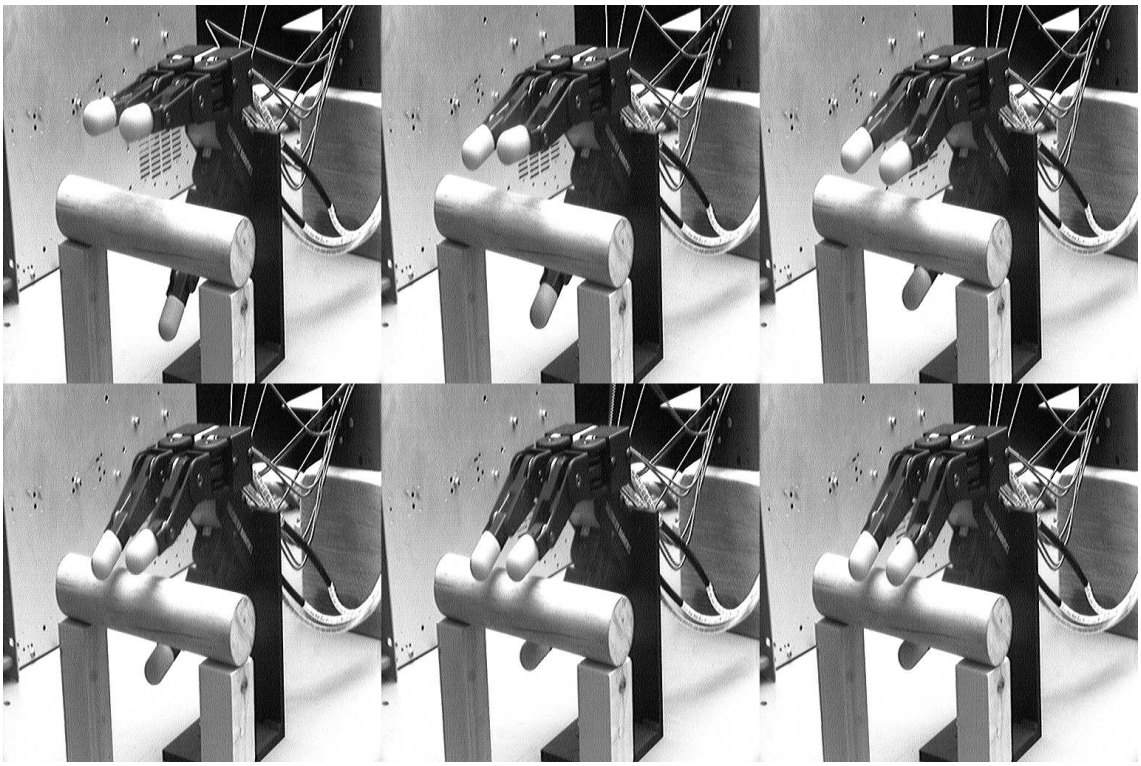


Figure 5.7: *La main de Salisbury en train de se configurer pour saisir un objet. La configuration géométrique est bonne. Le simulateur physique servira à vérifier si la saisie est dynamiquement stable.*

En fait pour la même stratégie de saisie, l'objet peut se comporter différemment. La figure 5.8 montre un modèle de la main articulée de Salisbury [76], en train de saisir un objet sphérique et un objet cylindrique. Quand la force de frottement entre la main et l'objet est très faible, l'objet glisse de la main (figure 5.8).

Pour obtenir une saisie stable il y a deux solutions. La première consiste à augmenter la force du frottement entre la main et l'objet (figure 5.9) et la deuxième consiste à prendre l'objet dans la paume de la main (figure 5.10). La stabilité de la saisie par la paume d'une main articulée est le résultat d'un grand nombre de contacts entre la main et l'objet à saisir, comme le montre la figure 5.10. Chaque contact introduit une force de collision et une force de frottement. Les deux types de forces causent une dissipation de l'énergie cinétique de l'objet et ralentissent, par conséquent, sa vitesse. Comme l'objet n'est pas capable, par définition, de sortir de la main, il finit par se stabiliser. Le modèle de la main de Salisbury comporte

[5]la pré-configuration est la configuration géométrique que la main doit prendre juste avant se fermer autour de l'objet. Cette configuration dépend de la forme de l'objet et des contraintes imposées par l'environnement

200 particules. Le pas de temps moyen pour réaliser la simulation était de 0.004 seconde. Le temps d'exécution mis pour simuler le mouvement de la main sur 50 secondes réelles était légèrement inférieur à 50 secondes.

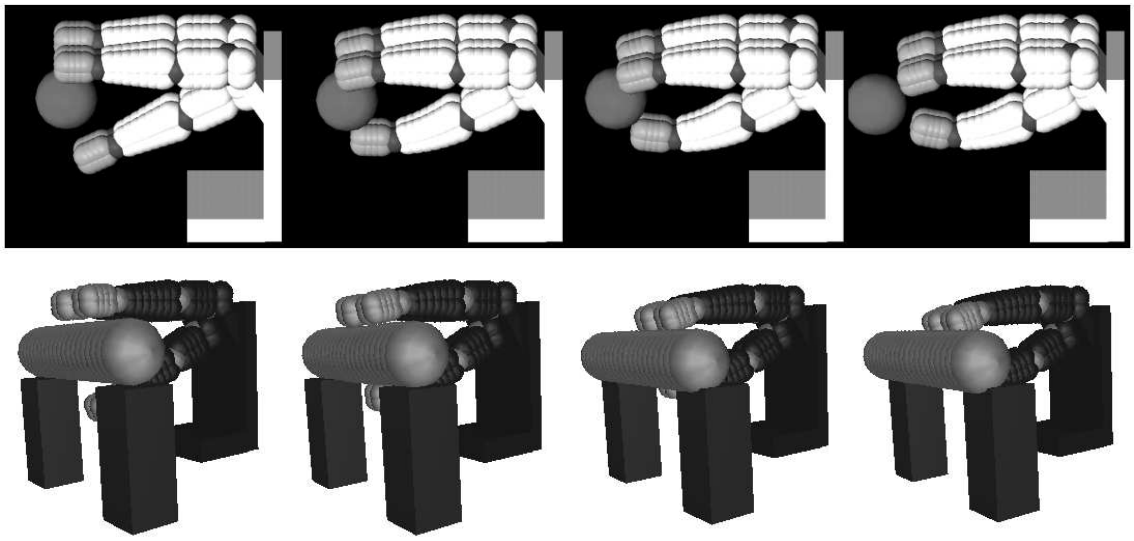


Figure 5.8: *Quand la force de frottement entre la main et l'objet à saisir est très faible, l'objet glisse hors de la main.*

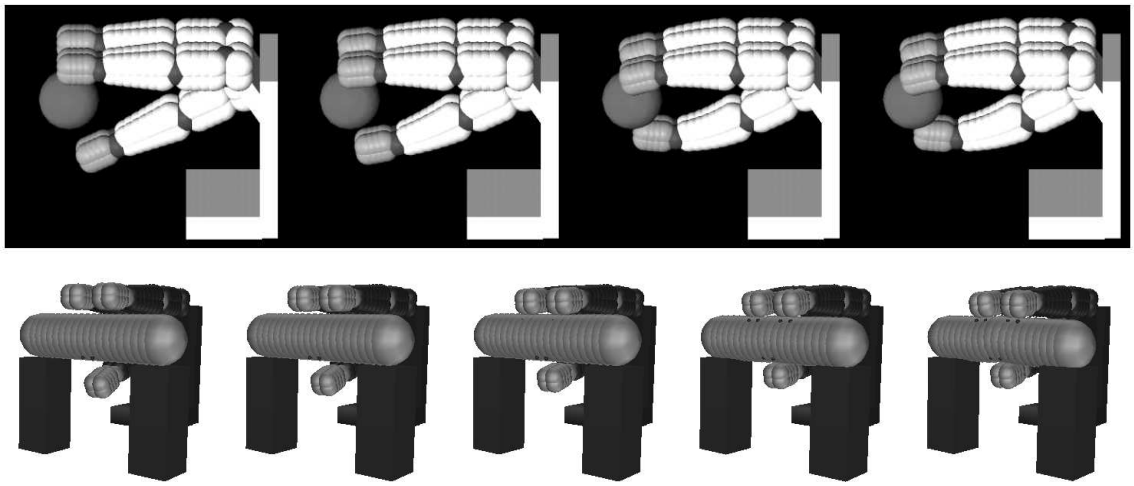


Figure 5.9: *Pour rendre la saisie plus stable on peut augmenter la force du frottement.*

Le mouvement de la main est contrôlé comme il est expliqué dans §5.2. On applique sur les différents doigts des forces capables de les joindre. Les positions des articulations sont automatiquement obtenues par le simulateur dynamique et ce sont ces valeurs qu'il faut envoyer à la vrai main articulée.

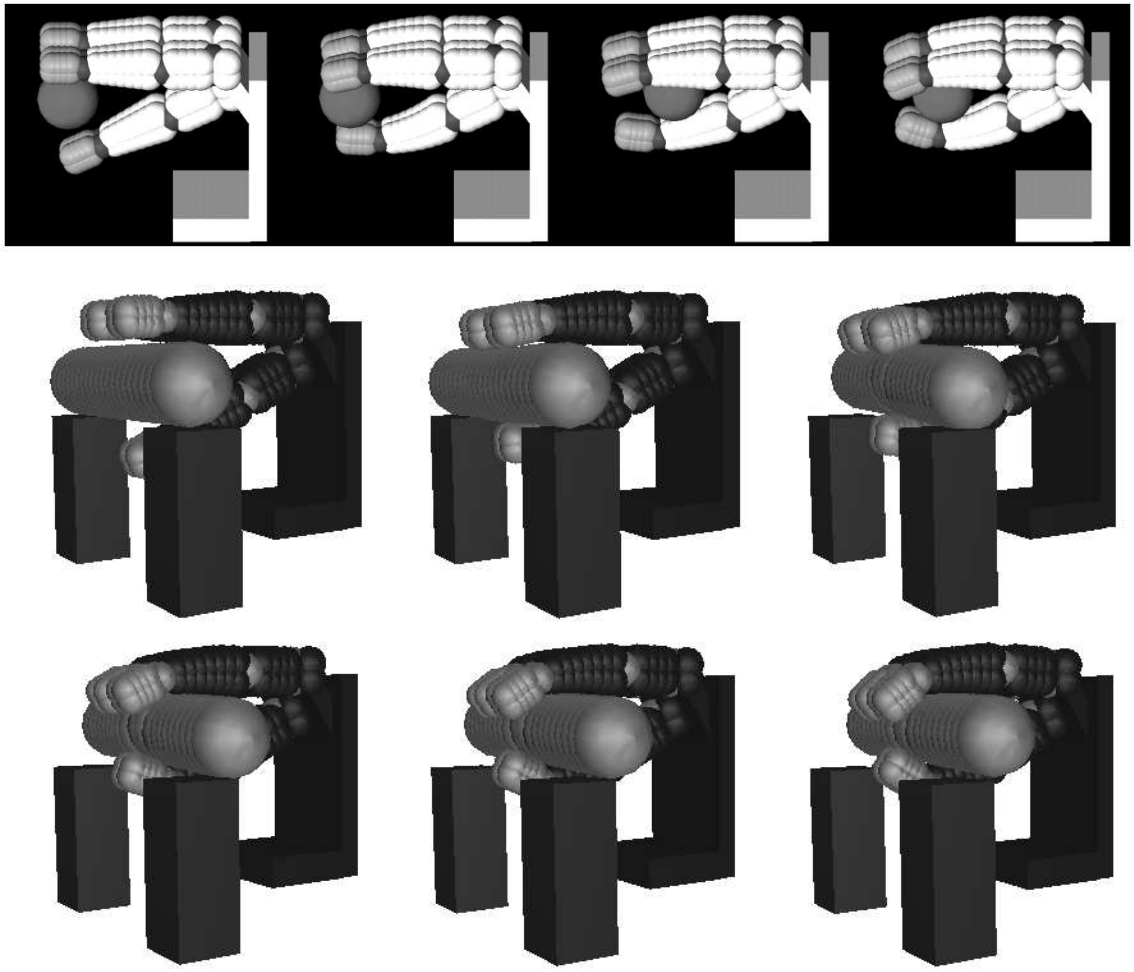


Figure 5.10: *La saisie par la paume de la main rend la saisie plus stable.*

La différence entre les différents cas de figures est la position où la force de contrôle est appliquée (figure 5.11). Le temps de simulation sur 50 secondes était d'environ 48 secondes.

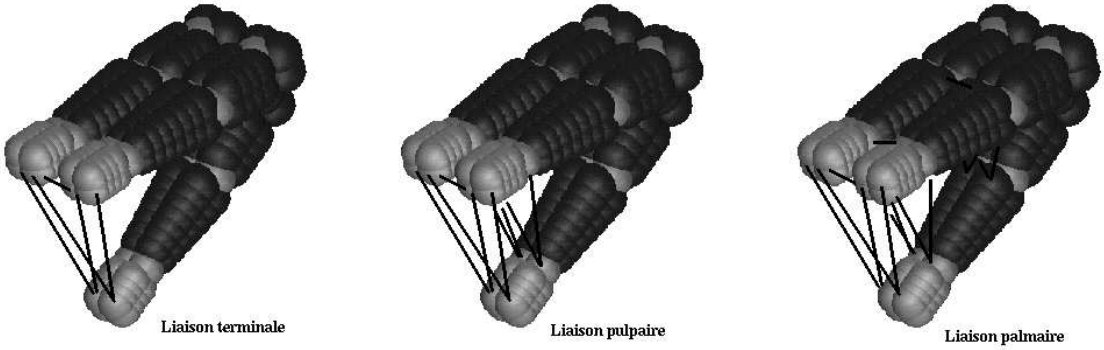


Figure 5.11: *Le mouvement de la main est contrôlé par des forces qui lient les différents doigts. Les points où les forces sont appliquées dépendent de la nature de la saisie. 1) saisie par les bouts des doigts 2) saisie pulpaire 3) saisie palmaire.*

5.5 Une application médicale: un ligament du genou

Ce travail a été réalisé en coopération avec le laboratoire TIMC Grenoble/FRANCE et le laboratoire de biomécanique de l'institut orthopédique Rizzoli Bologne/Italie.

La simulation du comportement dynamique de l'LCA (Ligament Croisé Antérieur) est très importants pour optimiser sa reconstruction après un accident et pour donner aux chirurgiens une idée plus claire sur la validité d'une position choisi pour mettre un ligament artificielle.

La biomécanique de LCA a été étudiée par plusieurs chercheurs qui ont proposé des différents modèles pour prédire le comportement de LCA sous certaines conditions[70, 15, 99, 12]. Ces modèles ne donnent pas des informations complètes sur l'LCA, et ils n'intègrent pas sa géométrie avec sa mécanique.

Notre travail a consisté à développer un modèle du ligament basé sur des données anatomiques et mécaniques plus précises et d'utiliser ce modèle pour simuler le comportement du ligament du genou pendant son mouvement passif [74, 75]. Ce modèle doit être capable d'évaluer au stade préopératoire l'effet de la position du ligament artificiel et sa tension initiale sur la cinématique du genou pendant son mouvement passif.

Méthodes et matériels

Pour construire un modèle dynamique du ligament, il nous faut acquérir sa forme géométrique et ses propriétés physiques (surtout l'élasticité). Pour cela quatre genoux de cochons ont été utilisés pour l'expérience. Le fémur est fixé tandis que le tibia est laissé mobile.

Au début, le mouvement passif a été manuellement enregistré entre la flexion totale et l'extension totale, en utilisant comme senseur digital $3D$ OPTOTRAK ou le bras de FARO (Fig.2.4). L'erreur de l'acquisition a été estimée en répétant le test plusieurs fois, sa valeur était d'environ $1mm$.



Figure 5.12: *Le mouvement passif du genou est exécuté par un chirurgien et la trajectoire est enregistrée par le bras de FARO*

Le tibia a alors été fixé en flexion totale, et le genou a été nettoyé autour du ligament pour le rendre visible. Dans ces conditions la surface de l'LCA a été digitalisée selon les directions des fibres (Fig.5.13). On a digitalisé 9 fibres sur la surface du ligament. (20 points pour chaque fibre).

Les positions des points sont exprimées dans un repère fixe attaché au fémur (Fig.5.14).

Pour construire un modèle anatomique précis de chaque fibre, les points enregistrés sur la fibre ont été interpolés par BSpline et on a choisi d'une manière régulière 20 points sur cette courbe (Fig.5.15)

La construction du modèle dynamique

Les 20×9 points sont utilisés comme positions des particules qui définissent le modèle dynamique du ligament.

On a lié chaque couple de particules successives appartenant à la même fibre par un connecteur linéaire, et chaque triplet de particules successives qui appartiennent à la même fibre par un connecteur angulaire dont l'élasticité est nulle. Le rôle



Figure 5.13: *Digitalisation des fibres du ligament quand elles sont en flexion complète*

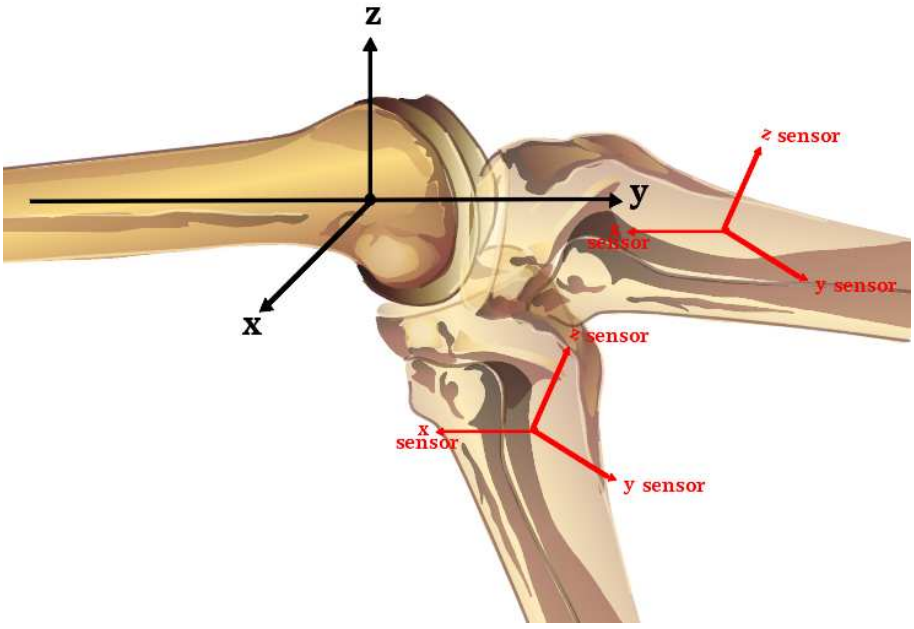


Figure 5.14: *Un repère fixe est lié au fémur et un repère mobile est lié au tibia. La figure montre la position de repère mobile dans les deux positions extrêmes du tibia.*

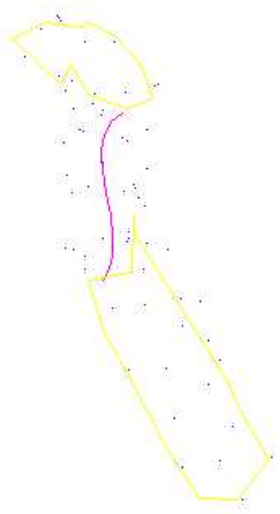


Figure 5.15: *On applique l'interpolation de Spline sur les points repérés sur chaque fibre afin de trouver sa forme exacte.*

de ces connecteurs angulaires est d'amortir la vitesse de chaque fibre pendant son mouvement.

Les particules voisines qui appartiennent à deux fibres voisines sont liées par des amortisseurs purs.

La constante d'élasticité des connecteurs linéaires dans une fibre est calculée en se basant sur des données mécaniques globales qu'on trouve dans la littérature [3, 97]. La viscosité de ces connecteurs n'a pas cependant été estimée dans la littérature. Cette viscosité n'a pas d'effet sur les résultats attendus car on mesure les élongations des fibres quand leurs vitesses sont nulles. Donc, la valeur de la viscosité a été déterminée empiriquement afin de faciliter le contrôle du ligament et d'obtenir un comportement géométrique correct. Le résultat était un modèle déformable totalement flexible, mais assez rigide en traction (Fig.5.16).

Le mouvement passif est exécuté en tirant (§5) le tibia vers les positions déjà enregistrées sur le genou réel. Le contrôle du tibia consiste à l'amener avec une faible vitesse ($< 0.1mm/s$) à chacune de ces positions.

Résultats:

Pour montrer l'exactitude du modèle et sa capacité à décrire le comportement dynamique réel du ligament, on a imposé au modèle de suivre la trajectoire réelle déjà enregistrée (figure 5.17). Pendant le mouvement du modèle, on a calculé la position du ligament, sa section, son volume, les longueurs des fibres et les forces aux points d'attachement. On a vérifié que les positions des fibres antérieures de l'LCA coïncidaient avec leurs positions réelles avec une erreur relative de 5%, ainsi que

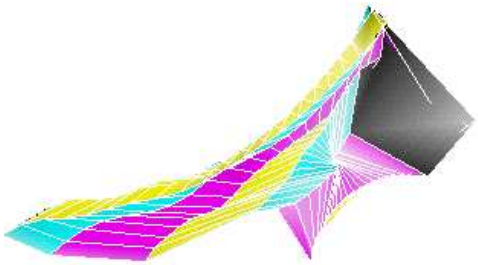


Figure 5.16: *Un modèle dynamique 3D du ligament. Le fémur est en haut. les fibres postérieures sont à gauche et les fibres antérieures sont à droite.*

les élongations des fibres avec les données reportées par les autres chercheurs. La comparaison concerne seulement certaines fibres qui peuvent être visibles pendant le mouvement passif, mais le modèle peut nous donner les positions et les élongations de toutes les fibres.

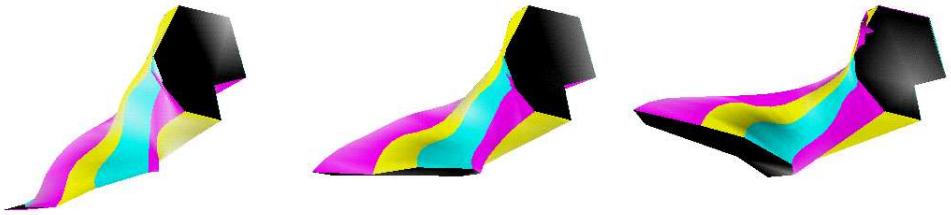


Figure 5.17: *Les déformations de l'LCA pendant le mouvement passif.*

La perspective à long terme de ce travail est de l'utiliser pour la planification chirurgicale. La précision de ce modèle le rend convenable pour une utilisation ultérieure afin d'optimiser le placement de l'LCA si les images MRI ou CT nous permettent d'identifier les surfaces des fibres.

5.6 Une application graphique: les effets spéciaux sur les images

Cette application est fait en coopération avec la société Getris-Images[6].

Le sujet de la coopération est l'intégration de notre simulateur dynamique (le système *Robot*Φ) avec la carte DVE (digital video effect) développée au sein de l'entreprise Getris-Images. La carte DVE permet de faire bouger des images $2D$ dans l'espace $3D$ et d'appliquer certains effets spéciaux géométriques sur ces images (vibration, onde, etc...) et de combiner certains effets ensemble (mouvement plus vibration par exemple).

But de l'intégration

Le but de cette intégration est de:

- enrichir les effets spéciaux que l'on peut obtenir par la carte DVE en créant un modèle dynamique de ces images qui peut être manipulé par des forces extérieures;
- simplifier l'interface utilisateur avec la carte DVE en permettant à l'utilisateur d'exprimer ses requêtes par une force appliquée sur son image au lieu d'être obligé de décrire carrément la nouvelle forme déformée de son image;
- aboutir à une nouvelle carte DVE capable de manipuler des objets $3D$ et non pas seulement des images $2D$. Cette carte va contenir une version câblée et parallèle de *Robot*Φ qui permettra probablement d'obtenir des performances "temps réel", ce qui est très important, à la fois pour les applications de la vidéo professionnelle et pour celles de la robotique.

Travail réalisé

Une version de *Robot*Φ a été adapté et muni d'une interface graphique sous le système d'exploitation *Windows*95 sur PC. Cette interface permet de définir des grilles $2D$ qui peuvent s'évoluer dans le $3D$ et de régler leurs propriétés physiques (élasticité, plasticité et viscosité). Le réglage de ces paramètres peut se faire d'une manière globale (s'il s'agit d'une grille homogène) ou d'une manière locale (s'il s'agit d'une grille non homogène). L'utilisateur peut exprimer ses requêtes par des forces à appliquer sur un des points de cette grille et le système donne l'effet de ces forces sur la grille. La carte DVE peut se servir de ces grilles déformables et de projeter sur elles n'importe quelle image. Cette carte est capable de faire le mapping en temps réel (figures 5.18 et 5.19).

Le travail va continuer pendant l'année 1996/1997 (dans le cadre d'un post-doctorat industriel INRIA) afin de construire une carte DVE $3D$ et une version parallèle de *Robot*Φ.

[6] La société Getris-Images conçoit, développe et commercialise des outils matériels et logiciels pour l'informatique graphique. ZIRST de Grenoble, France

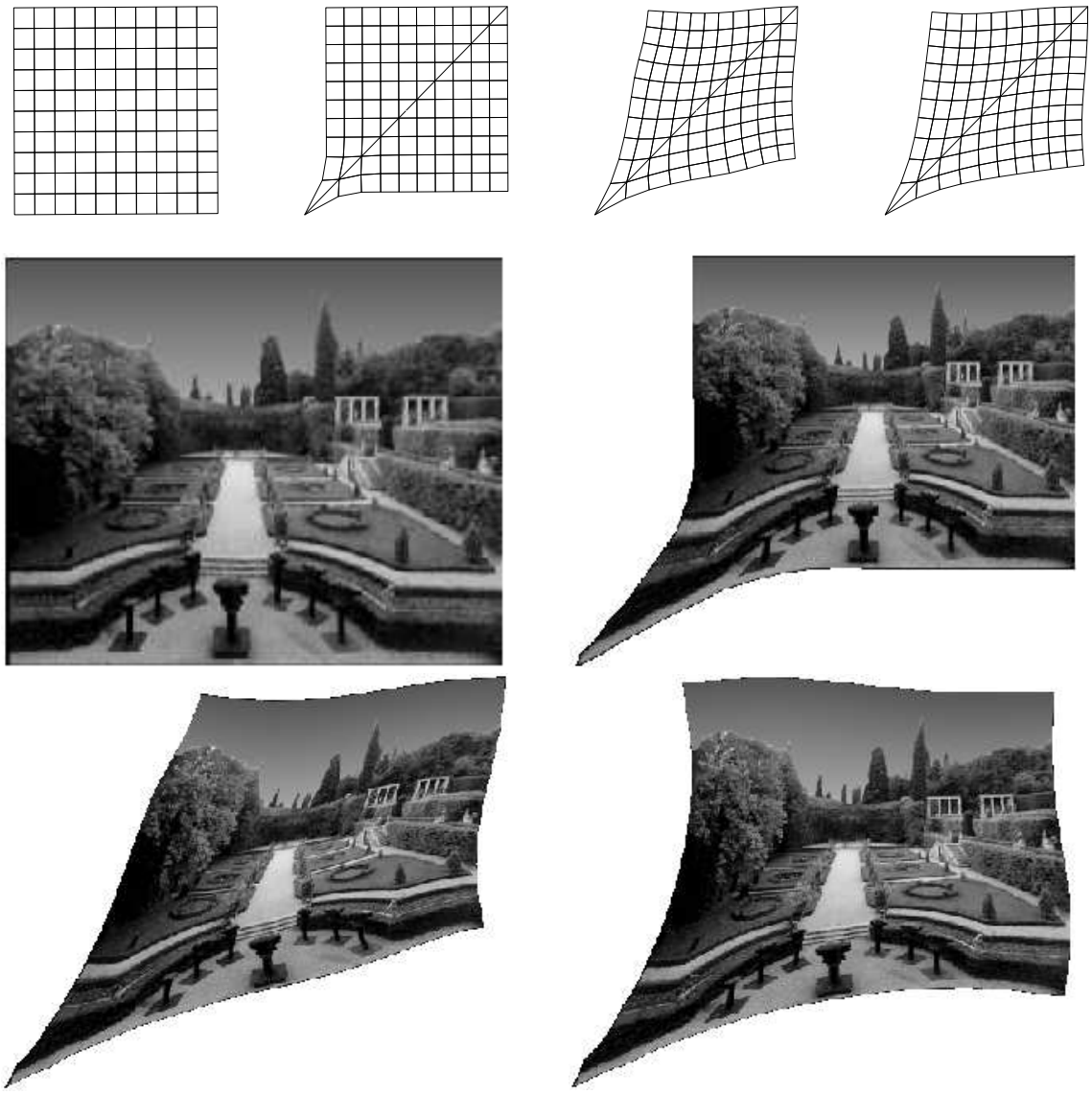


Figure 5.18: *Dans cet exemple la force est appliquée sur le coin de l'image et elle tire vers le bas/gauche. Donc l'image bouge tout en se déformant. Là haut on voit la grille générée par Robot*Φ *et en bas on voit la projection d'une image sur cette grille.*

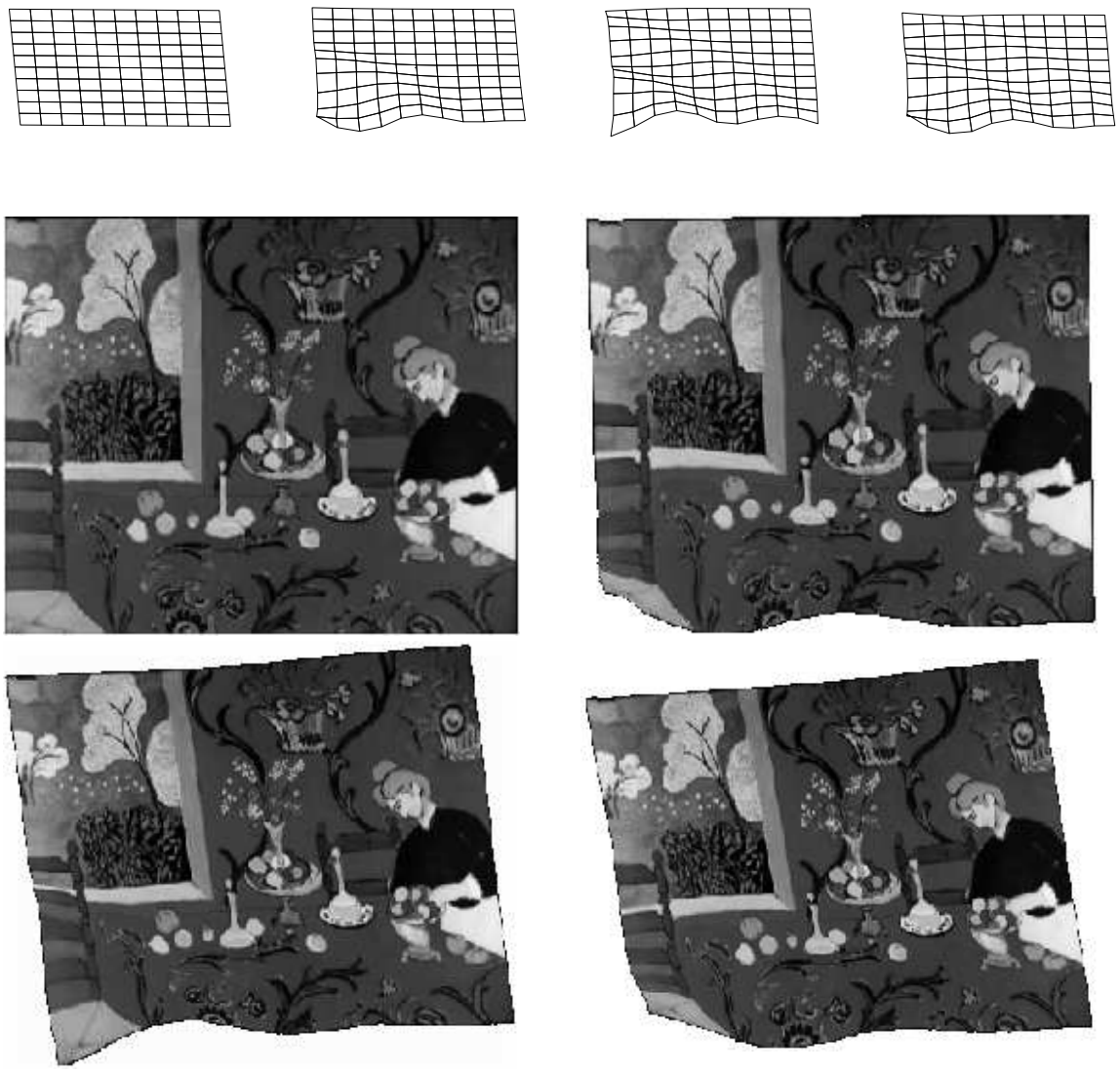


Figure 5.19: *Dans cet exemple la force est appliquée sur le coin de l'image. cette force est une combinaison de deux forces. Une qui tend à tirer l'image vers le bas/gauche et l'autre tend à vibrer l'image autour de son plan initial. Là haut on voit la grille générée par RobotΦ et en bas on voit la projection d'une image sur cette grille.*

5.7 Conclusion

La simulation dynamique permet de contrôler les objets par l'application de forces virtuelles. Ces forces ne sont pas le résultat d'un actionneur, donc elles peuvent être appliquées à n'importe quel endroit de l'objet. Le simulateur dynamique permet à ces forces de se propager à l'intérieur de l'objet jusqu'aux articulations et actionneurs et de les reconfigurer en fonction du mouvement. La reproduction de ces configurations en fonction du temps, peut (sous condition de non-glissement) générer le même mouvement.

Donc le simulateur est un environnement virtuel qui peut servir à tester les nouveaux algorithmes avant de les appliquer sur des matériaux réels. Cela peut donc servir à éviter les accidents potentiels qui peuvent être causés par une faute de programmation.

Notre simulateur dynamique a été utilisé dans différents domaines d'applications (robotique, médical et graphique).

Il a été utilisé pour simuler la stabilité de la saisie par la main de Salisbury. Ceci peut servir à valider les différentes stratégies de saisie données par un planificateur géométrique. Nous avons eu des bonnes résultats qualitatifs mais nous n'avons pas encore obtenu des résultats quantitatifs.

Le comportement du ligament du genou pendant le mouvement passif a été, également, simulé. Nous avons pu obtenir des résultats quantitative et nous avons pu engendré par simulation la trajectoire enregistrée pendant la phase d'acquisition des données à une erreur relative de 5%.

Une application graphique (en coopération avec la société Getris-Images) consiste à utiliser le système pour générer des effets spéciaux sur des images $2D$ qui évoluent dans le $3D$. Cette application va continuer afin de créer une version câblée de $Robot\Phi$ que l'on espère qu'elle tourne en temps réel.

Chapter 6

Identification

6.1 Introduction

Avec le système *Robot*Φ on peut créer un modèle paramétré d'un objet et simuler son mouvement, ses déformations et ses interactions avec son environnement. Le comportement obtenu dépend des paramètres physiques qui définissent le modèle, telle que l'élasticité, la plasticité, la viscosité, la rigidité/viscosité de la collision, la masse, et la distribution de cette masse. L'identification consiste à trouver les valeurs de ces paramètres pour lesquelles le comportement simulé coïncide avec le comportement réel de l'objet de référence. Dans ce chapitre on divise le problème de l'identification en deux parties:

- l'identification des propriétés de l'inertie qui consiste à trouver une distribution de la masse de l'objet entre les différentes particules afin de garder le même centre d'inertie et la même matrice d'inertie que l'objet réel;
- l'identification des paramètres physiques du modèle: élasticité, plasticité, viscosité, rigidité et viscosité de la collision.

6.2 La conservation des propriétés d'inertie

La conservation de propriétés d'inertie [93, 54] consiste à trouver la position p_i et la masse m_i à donner à chaque particule i, afin de conserver la matrice d'inertie et le centre d'inertie de l'objet référence.

6.2.1 Formulation du problème

Le centre d'inertie P_c d'un objet est donné par l'équation suivante:

$$P_c = \frac{1}{M} \iiint_{volume} P \, \delta m$$

où M est la masse totale de l'objet. La matrice d'inertie est donnée par l'équation suivante:

$$\begin{pmatrix} A & -F & -E \\ -F & B & -D \\ -E & -D & C \end{pmatrix} = \begin{pmatrix} \iiint (y^2+z^2)\,\delta m & -\iiint x\,y\,\delta m & -\iiint x\,z\,\delta m \\ -\iiint x\,y\,\delta m & \iiint (x^2+z^2)\,\delta m & -\iiint y\,z\,\delta m \\ -\iiint x\,z\,\delta m & -\iiint y\,z\,\delta m & \sum (x^2+y^2)\,\delta m \end{pmatrix}$$

où $(x\,y\,z)$ représente la position d'un point P de l'objet. Pour passer à une représentation discrète, il faut remplacer l'intégrale $(\int)$ par une somme finie $\sum$. L'identification revient alors à "identifier" les deux dernières équations dans l'espace continu et dans l'espace discret, comme suit :

$$\begin{pmatrix} A & -F & -E \\ -F & B & -D \\ -E & -D & C \end{pmatrix} = \begin{pmatrix} \sum ({y_i}^2+{z_i}^2)\,m_i & -\sum x_i\,y_i\,m_i & -\sum x_i\,z_i\,m_i \\ -\sum x_i\,y_i\,m_i & \sum ({x_i}^2+{z_i}^2)\,m_i & -\sum y_i\,z_i\,m_i \\ -\sum x_i\,z_i\,m_i & -\sum y_i\,z_i\,m_i & \sum ({x_i}^2+{y_i}^2)\,m_i \end{pmatrix}$$

$$P_c = \frac{1}{M}\sum_{i=1}^{n} m_i\,P_i$$

Le système d'équations donné ci-dessus comporte 9 équations (trois pour le centre d'inertie et six pour la matrice d'inertie), et $4n+1$ inconnues (la masse m_i et la position $P_i = (x_i, y_i, z_i)$ des n particules plus n qui est aussi inconnu).

Dans le cas général, ce système peut avoir une infinité de solutions. De plus le système n'est pas linéaire et il n'y a pas de méthodes qui permettent de calculer une solution de ce système.

6.2.2 Solution

Pour simplifier le problème il faut diminuer le nombre d'inconnues et rendre le système linéaire. Pour le rendre linéaire il faut fixer, intuitivement, les positions et le nombre des particules. Il reste donc les masses m_i comme n inconnues. Si n est supérieur au nombre d'équations (9) le système aura une infinité de solutions. Quand un objet se déforme, une solution donnée peut ne plus être valide. Pour que la solution obtenue soit toujours valide il faut respecter la matrice d'inertie pas seulement globalement mais aussi localement.

La méthode de la discrétisation automatique proposé dans §2.3.3, ne décompose pas seulement l'objet en plusieurs particules mais aussi en plusieurs hexaèdres. Chaque hexaèdre a, par définition, 8 facettes et 8 sommets (figure 6.1). Pour respecter localement les propriétés d'inertie de l'objet, il faut résoudre le système pour chaque hexaèdre. Le système d'équation n'a pas de solution pour 8 particules qui se trouvent sur la surface d'un hexaèdre, car cela nous oblige à surestimer la valeur de la matrice d'inertie. Pour être sûr que la solution existe, il faut ajouter une $9^{\text{ième}}$ particule au centre d'inertie de chaque hexaèdre (figure 6.1).

Ce choix intuitif des positions des particules n'a pas donné une solution unique. Parfois la solution peut ne pas exister. Pour cela on a simplifié davantage le problème en décomposant chaque hexaèdre en plusieurs tétraèdres. Le choix intuitif était de mettre quatre particules aux sommets du tétraèdre et une particule à son centre d'inertie (figure 6.2). Ce choix nous a donné une solution[1] unique du système qui

[1] la solution est obtenue en utilisant MAPLE

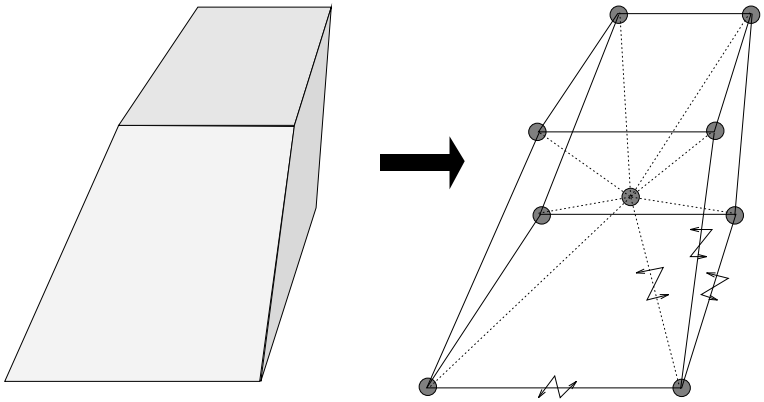


Figure 6.1: *La discrétisation automatique décompose un polyèdre en plusieurs hexaèdres, chacun est défini par* 8 *particules.*

consiste à répartir la masse du tétraèdre entre les cinq particules de la manière suivante :

$$m_i = \begin{cases} \frac{1}{20}M & i \text{ si la particule se trouve sur un sommet} \\ \frac{4}{5} * M & i \text{ si la particule se trouve au centre d'inertie du tétraèdre} \end{cases}$$

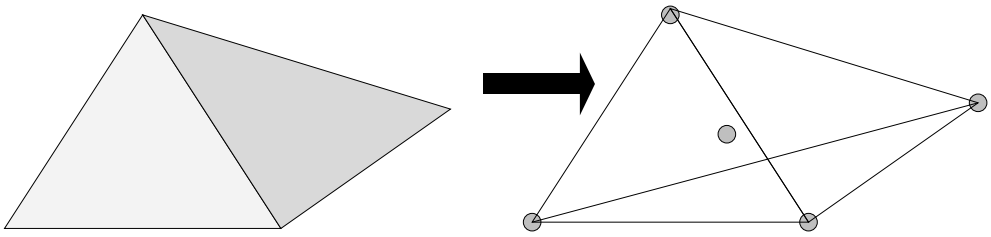


Figure 6.2:

Donc, pour respecter les propriétés de l'inertie, on peut décomposer chaque hexaèdre en plusieurs tétraèdres, puis on remplace chaque tétraèdre en cinq particules comme expliqué ci-dessus.

Le choix du nombre et des positions des tétraèdres à l'intérieur de l'hexaèdre dépend de deux critères:

- le nombre final des particules nécessaires pour représenter l'hexaèdre: il faut, au moins, ajouter une particule supplémentaire au centre d'inertie de chaque tétraèdre. Si on diminue le nombre de tétraèdres, il y aura économie au niveau du nombre de particules;
- les positions de ces tétraèdres affectent aussi le comportement de l'objet. S'ils ne sont pas "symétriques", cela donne un comportement non isotrope de l'objet en question, sauf quand l'objet est parfaitement rigide.

Nous avons développé trois méthodes pour faire cette décomposition en tétraèdres:

Une méthode symétrique et exacte: Dans ce cas, on commence par ajouter *une* particule au centre d'inertie du tétraèdre, puis une particule au centre d'inertie de chaque facette (ce qui fait 6 particules au total). Chaque facette, peut donc être vue comme quatre triangles, chacun est formé d'une arête de la facette et de son centre d'inertie. Chaque triangle forme un tétraèdre avec le centre d'inertie de l'hexaèdre. Cela donne 24 tétraèdres. Il faut ajouter une particule au centre d'inertie de chaque tétraèdre (ce qui fait 24 particules au total). le nombre total de particules devient alors $8 + 1 + 6 + 24 = 39$. Cette méthode donne un comportement isotropique, et il donne la valeur exacte de la matrice d'inertie de l'hexaèdre originale, mais il nécessite un grand nombre de particules (31 particules supplémentaires pour 8 particules au départ, ce qui fait 79.55%). Pour des objets très déformables cette méthode donne de bons résultats.

Une méthode non-symétrique et exacte: Dans ce cas, on divise l'hexaèdre en 5 tétraèdres, dont tous les sommets sont partagés avec les sommets de l'hexaèdre. En ajoutant une particule au centre d'inertie de chacun d'entre eux, on obtient un nombre total de 13 particules (38.5% seulement de particules sont supplémentaires). En générale on utilise cette méthode pour discrétiser les objets rigides.

Une méthode symétrique et approximative: Dans ce cas, on procède comme dans le premier cas, mais on élimine les particules qui se trouvent au centre d'inertie de chaque tétraèdre, en divisant ses masses entre les sommets du tétraèdre en question tout en gardant le même centre d'inertie de chaque tétraèdre. Le centre d'inertie peut être respecté si on donne à chaque sommet le quart de la masse du centre d'inertie. Puis on élimine le centre d'inertie de chaque facette en divisant ses masses entre les sommets de cette facette, tout en gardant le même centre d'inertie. L'hexaèdre dans ce cas est représenté par 9 particules dont le centre d'inertie est le même que l'hexaèdre d'origine, mais ils ont une valeur approximative de la matrice d'inertie. Pour améliorer davantage cette précision, on enlève une partie de la masse de chaque sommet et on la met au centre d'inertie. Cette partie est calculée de manière à être proportionnelle à la masse de chaque sommet. Le pourcentage idéal qui nous fait approcher le mieux de la vraie valeur de la matrice d'inertie peut être trouvé d'une manière récursive. Pour 10000 tests, la valeur moyenne de la précision relative de cette méthode était d'environ 0.0003. L'avantage de cette approche réside dans le fait que l'on représente un hexaèdre par seulement 9 particules (11% de particules supplémentaires). Actuellement, c'est cette méthode que l'on utilise.

L'approche de l'identification des propriétés de l'inertie a un aspect adaptatif, car le nombre d'hexaèdres peut être déterminé en fonction de la déformabilité de l'objet. Plus l'objet est déformable, plus la discrétisation doit être fine. La différence entre la matrice d'inertie de l'objet réel et la matrice d'inertie de l'objet modèle, change pendant les déformations de l'objet. 10000 tests ont montré que si on change la position relative d'une particule de 10% par rapport à son hexaèdre, l'erreur relative

change de 0.1%. On peut mesurer cette erreur pour des déformations maximales de l'objet et on choisit le nombre d'hexaèdres pour que l'erreur relative maximale soit inférieure à un seuil donné.

6.3 Identification des paramètres physiques

L'identification des paramètres physiques (élasticité, plasticité, etc...) consiste à trouver les bonnes valeurs de ces paramètres pour lesquels le comportement dynamique de l'objet modèle ressemble à celui de l'objet réel.

En fait le comportement d'un objet i est défini par une fonction complexe $\vec{C_t} = f(\vec{Fs}, Pr1, Pr2, .., Pr_i, ..Pr_n)$, où $\vec{C_t}$ est la configuration (vitesse, position, déformation) de l'objet à l'instant t ,$\vec{F_s}$ sont les forces extérieures appliquées sur cet objet (Ces forces dépendent de la position de l'objet, de sa vitesse et des positions des autres objets) et Pr_i est le paramètre i qui définit les comportements intrinsèques de l'objet (élasticité, plasticité, etc).

L'identification donc consiste à trouver la fonction inverse f^{-1} de la fonction f:

$$(Pr_1, Pr_2, ..., ..Pr_i, ..Pr_n) = f^{-1}(C\vec{V}_t, \vec{Fs})$$

et dans ce cas $C\vec{V}_t$ devient le comportement que l'on veut imposer à l'objet.

Il est évident qu'on ne peut pas inverser cette fonction à cause de sa non-linéarité et de sa discontinuité (§ 3), donc le problème de l'identification devient un problème d'optimisation de paramètres. Alors il nous faut chercher les paramètres $Pr_1, ..Pr_n$ pour que la fonction $C\vec{V}_t - \vec{C_t}$ soit minimale. Dans le cas général on ne connaît pas la valeur qui minimise la fonction à optimiser, mais dans notre cas particulier, on sait que cette valeur, qui est la différence du comportement, doit converger vers zéro.

Plusieurs méthodes existantes permettent d'optimiser une fonction. On peut distinguer entre deux classes de méthode: les méthodes dépendantes du modèle et les méthodes indépendantes du modèle:

Les méthodes dépendantes du modèle: La méthode de descente de gradient [79] permet de trouver un minimum local d'une fonction quelconque à condition de pouvoir la dériver numériquement ou analytiquement. Dans notre cas il est impossible de dériver la fonction, car elle est fortement discontinue, et on ne dispose pas d'un formalisme analytique de cette fonction. Le programme qui calcule cette fonction contient plus de 10000 lignes de code.

Les réseaux de neurones ont déjà été utilisés pour trouver les paramètres physiques d'un objet déformable modélisé par le système masse/ressort [88]. L'inconvénient de cette méthode réside dans le fait qu'il faut construire un réseau de neurones pour chaque objet, ce qui fait de l'identification une tâche lourde. De plus le temps nécessaire pour identifier les paramètres est relativement long par cette méthode.

Les méthodes indépendantes du modèle: Ces méthodes considèrent que le modèle est une boite noire dont la sortie (dans notre cas la différence entre le

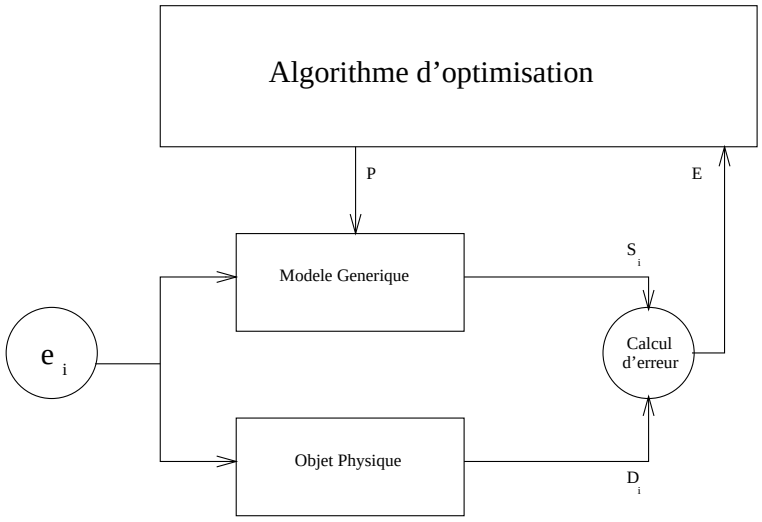


Figure 6.3: *Représentation graphique du comportement des méthodes indépendantes du modèle*

comportement réel et celui donné par le système) dépend des valeurs des paramètres physiques (figure 6.3).

La recherche exhaustive est une méthode directe qui consiste à discrétiser l'espace des paramètres et à essayer toutes les combinaisons possibles pour choisir la meilleure solution. Cette méthode peut être utile quand l'espace des paramètres est très réduit, ce qui n'est pas notre cas.

Le recuit simulé [31] consiste à mettre à jour une solution en imitant le comportement d'une particule agitée par la température. Quand la température est élevée la particule aura suffisamment d'énergie pour atteindre des configurations éloignées de sa position de départ. Si la température baisse, la particule se stabilise dans le minimum local le plus proche. Pour éviter de tomber dans un minimum local, il faut augmenter la température du système ce qui peut ralentir la convergence. En pratique cette méthode tombe facilement dans des minima locaux.

Les algorithmes génétiques [40] (voir §6.3.1) sont plus robustes, et ils ont déjà donné des résultats intéressants [48]

6.3.1 Les algorithmes génétiques: principe de base

Les algorithmes génétiques sont des méthodes basées sur l'évolution d'une espèce vivante. Etant donnée une famille de solutions initiales (individus) choisies au hasard, cette approche consiste à faire évoluer cette famille de solutions tout en gardant un nombre constant d'individus et jusqu'à ce que tous les individus soient identiques. Chaque solution est représentée par une chaîne (génôme) de bits (gène). L'évolution est basée sur trois règles:

- la sélection: elle consiste à choisir les meilleurs génômes et à enlever les autres. Le pourcentage des génômes à choisir est un paramètre de l'approche. Une

solution est meilleure si elle minimise davantage la fonction en question (la fonction à optimiser);

- la mutation: elle consiste à modifier au hasard la valeur d'un ou plusieurs gènes du génôme;
- le croisement: le croisement de deux individus consiste à générer un nouvel individu (génôme) dont les gènes sont hérités de ses deux parents. La méthode la plus utilisée pour croiser deux génômes consiste à concaténer le préfixe de l'un avec le suffixe correspondant de l'autre (figure 6.4).

L'évolution d'une génération pour obtenir une nouvelle génération, consiste à sélectionner les meilleurs individus pour qu'ils appartiennent à la nouvelle génération et à ajouter à cette nouvelle génération les individus qui sont les résultats de la mutation et du croisement des individus sélectionnés (figure 6.4).

La convergence est assurée par la sélection qui fait que la meilleure solution de la nouvelle génération est meilleure ou égale à la meilleure solution dans la génération précédente.

La convergence est caractérisée par une famille de solution dont tous les individus sont identiques, car à partir du moment où la solution optimale appartient à une génération, la mutation et le croisement ne peuvent plus générer de meilleure solution. Donc, seules les solutions qui sont identiques à la meilleure solution peuvent être sélectionnées.

6.3.2 Application sur le modèle dynamique

Malgré la généralité du formalisme de base des algorithmes génétiques, ce formalisme pose certains problèmes quand on l'applique sur l'identification des paramètres physiques pour lesquels on proposera des solutions:

Manque du sens physique

Le codage d'une solution par une chaîne de bits ne représente pas le sens physique de cette solution et la mutation peut donner des solutions incohérentes. Par exemple l'élasticité d'un objet est une valeur positive et on ne peut pas garantir que la mutation au niveau de bits ne donnera pas des valeurs négatives. Dans ce cas la fonction d'évaluation risque de provoquer des erreurs. Pour éviter ce problème et pour garder un sens physique à chaque gène, on représente une solution (génôme) par un vecteur de réel (gènes) ou chaque réel représente la valeur d'un paramètre. La valeur de chaque paramètre varie dans un intervalle borné qui définit son domaine de définition. La mutation d'un gène consistera à ajouter une valeur réelle à sa valeur tout en restant dans le domaine de définition de ce paramètre.

Les doubles

Le formalisme de base accepte qu'une solution se répète. Cela nous oblige à évaluer cette solution plusieurs fois et à converger très lentement. Dans le cas d'un simulateur dynamique l'évaluation d'une solution peut prendre un temps important (5 ou 10 minutes). La manipulation des doubles, risque de ralentir

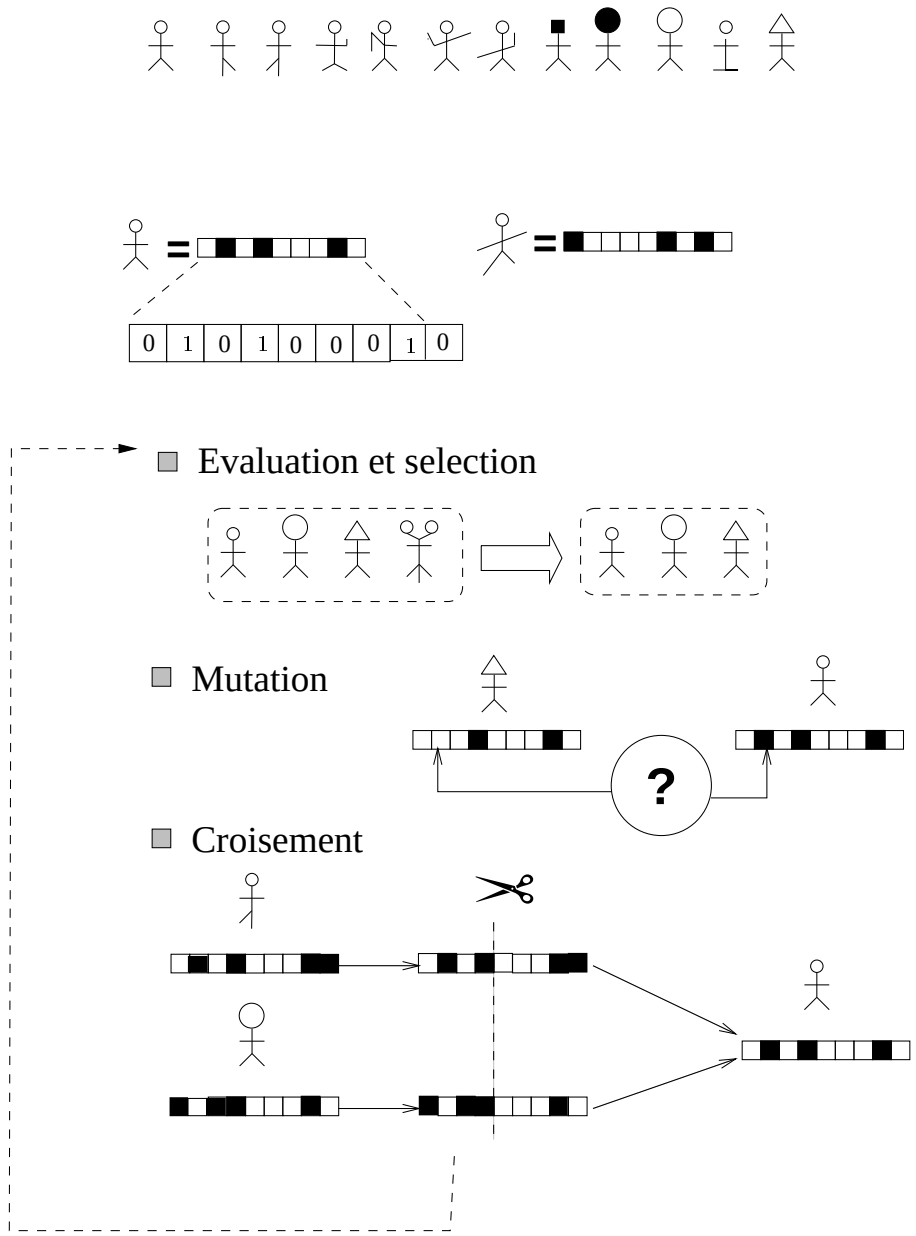


Figure 6.4: *A partir d'une famille initiale de solutions, les algorithmes génétiques génèrent d'autres familles en utilisant trois actions: sélection, mutation et croisement. La convergence est caractérisée par une famille dont les individus sont tout identiques*

énormément le temps d'exécution. Pour cela quand une solution se répète, on enlève automatiquement tous les autres clones sans les évaluer. Comme le rôle des doubles est de caractériser la convergence, il nous faut dans ce cas le caractériser autrement. Dans notre cas on ne cherche pas n'importe quelle solution qui minimise l'erreur $C_t - CV_t$ mais on cherche à rendre cette valeur minimale le plus proche possible de zéro. Dans le cas général, cette valeur est inconnue. Pour cela, la convergence est caractérisée simplement par une solution dont l'efficacité est inférieure à un certain seuil.

Une seule fonction d'évaluation

La sélection se fait selon la valeur donnée par la fonction d'évaluation. Dans le cas du système dynamique on ne peut pas caractériser le comportement d'un objet par une valeur unique et il nous faut plusieurs critères d'évaluation et non pas un seul. On distingue trois cas:

- les critères sont indépendants avec un ordre de priorité: Quand les critères sont indépendants, la satisfaction d'un critère n'affecte pas la satisfaction d'autres critères. Par exemple on veut que la différence de volume entre l'objet réel et son modèle soit inférieure à un certain seuil donné, mais dans ce cas on a une infinité de solutions. Parmi ces solutions on choisit la solution pour laquelle l'exécution est la plus rapide. Donc, la sélection se fait selon le critère le plus important (l'erreur du comportement). Si deux génômes vérifient ce critère, la sélection se fera selon le deuxième critère (la vitesse d'exécution), etc...;
- les critères sont indépendants sans ordre de priorité: Dans ce cas on peut choisir n'importe quel ordre et le traitement se fait comme dans le cas précédent;
- les critères sont dépendants: Dans ce cas on ne peut pas imposer un ordre de priorité car on n'est pas sûr que la solution existe. Par exemple on cherche les paramètres physiques d'un contrôleur capable d'amener le robot à une position donnée avec une vitesse donnée pendant une période limitée. Pour arriver plus vite il faut appliquer une force plus importante ce qui augmente la vitesse et cause un dépassement de la position. Si on applique une force moins importante, le robot arrive à cette position trop tard. Dans ce cas, le critère d'évolution devient une combinaison linéaire de ces trois critères. Le poids de chaque critère dépend de son importance.

Convergence lente

Le croisement ne fait que donner des nouvelles combinaisons entre les différents paramètres. C'est la mutation qui nous permet de trouver des nouvelles valeurs et de converger vers la solution optimale. Cette mutation se fait en ajoutant une valeur quelconque v à la valeur initiale du paramètre. Si la valeur v est fixe la précision de la valeur optimale de ce paramètre est d'environ $\frac{v}{2}$. Si on diminue cette valeur pour améliorer la précision, le temps d'exécution peut être considérablement augmenté. L'ajout d'une valeur aléatoire tiré dans un intervalle fixe $[a..b]$ n'améliore pas beaucoup le comportement de l'algorithme.

Quand la distance entre la valeur optimale du paramètre et sa valeur actuelle est trop petite par rapport à la largeur de cet intervalle, la possibilité d'ajouter une petite valeur devient minime. Donc, dans ce cas l'algorithme génétique converge très vite jusqu'à un certain voisinage de la solution optimale, puis il passe beaucoup de temps à faire le reste.

Notre solution [32] consiste à changer la largeur de cet intervalle en fonction de l'erreur commise en comportement. Pour cela on utilise une loi Gaussienne où la valeur aléatoire à ajouter x vérifie que $y = e^{\frac{(x-a)^2}{-2\mu}}$, où y est une variable aléatoire entre [0..1], a est l'espérance de la loi Gaussienne qui correspond, dans notre cas, à la valeur actuelle d'un paramètre, et μ est la variance de la Gaussienne qui définit un intervalle de valeurs les plus probables (figure 6.5). Donc la valeur x à ajouter doit être donnée par $x = a \pm \sqrt{-2.\mu.log(y)}$.

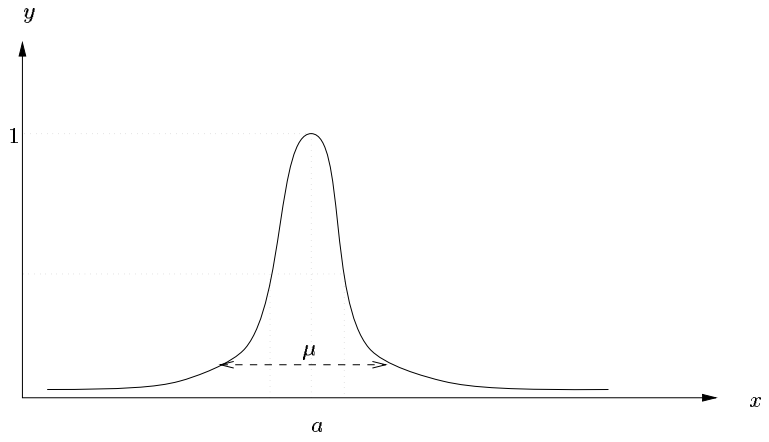


Figure 6.5: *La loi gausienne*

La figure 6.6, montre comment progresse la Gaussienne en approchant de la solution. Au commencement de la recherche, le Gaussienne est large, et permet donc à x d'atteindre n'importe quelle valeur de l'ensemble de définition, avec des probabilités à peu près égales. Au fur et à mesure de la recherche, l'erreur décroît, et donc la largeur de la Gaussienne aussi, réduisant les chances pour x d'atteindre des valeurs éloignées. La Gaussienne se déplace également sur la variable a qui doit muter, ainsi, la valeur de x sera proche de a. *La largeur de la Gaussienne diminue donc au fur et à mesure que l'on s'approche de la bonne solution, augmentant ses chances d'être mise en valeur.* L'inconvénient de cette approche dans le cas général est caractérisé par le risque de tomber dans des minima locaux, et le comportement de l'algorithme génétique dans ce cas s'approche de celui de la descente de gradient. Dans notre cas particulier, cela ne se passe pas car tant que l'erreur existe, la largeur de la Gaussienne n'est pas nulle. Cette approche nous a permis d'améliorer sensiblement la convergence de l'algorithme génétique. Un exemple qui nécessitait 200 générations pour converger, nécessite avec cette approche 20 générations seulement.

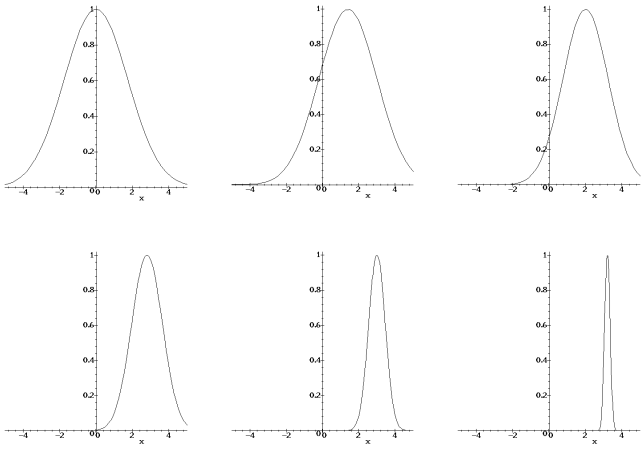


Figure 6.6: *L'évolution de la Gaussienne durant la recherche de la valeur x=3.4(maple).*

6.3.3 Expérimentation

Pour tester l'algorithme d'identification, on l'utilise pour trouver les paramètres avec lesquels le modèle se comporte d'une manière déjà définie. Plusieurs tests ont été faits:

Traitement d'un seul critère

Dans ce test on veut comparer le comportement analytique de deux masses liées par un ressort linéaire (figure 6.7) avec le comportement donné par *Robot*Φ.

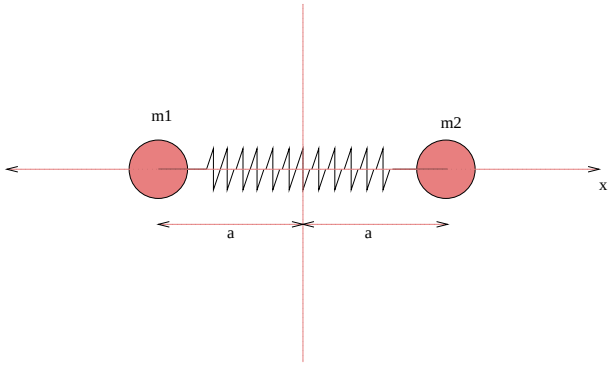


Figure 6.7: *Deux masses liées par une relation linéaire.*

Le comportement réel est analytiquement trouvé en utilisant MAPLE et pour des valeurs fixes de la rigidité du ressort $\lambda = 8.94$ et sa viscosité $\mu = 0.6$. La fonction d'évaluation sera donc la différence entre la trajectoire donnée par MAPLE et celle donnée par *Robot*Φ. En commençant par des valeurs aléatoires, les valeurs données par l'algorithme génétique au bout de 234 générations étaient $\lambda = 8.85$

et $\mu = 0.6425$ pour une erreur moyenne d'environ $4.9e^{-5}$. La figure 6.8 montre l'évolution de l'erreur en fonction du nombre de générations. On remarque que l'algorithme converge et qu'il existe des périodes de stabilisation. Ces périodes de stabilisation sont beaucoup plus longues si on n'utilise pas une loi Gaussienne.

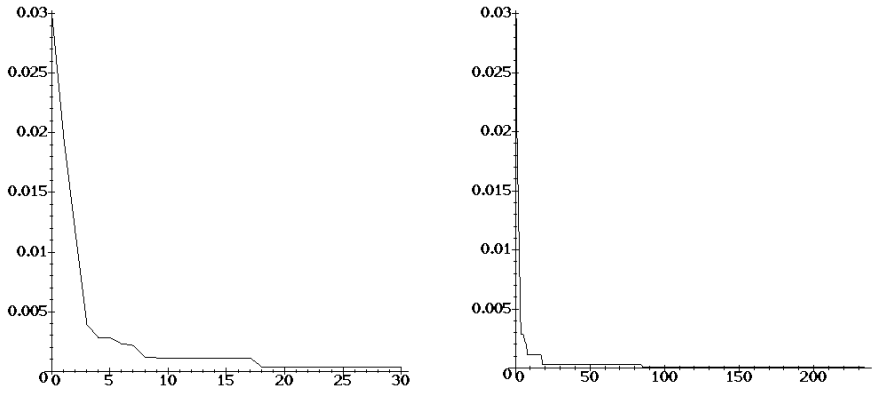


Figure 6.8: *L'évolution de l'erreur en fonction du nombre de générations.*

Traitement de deux critères indépendants

Identification solide/déformable: Le but de ce test est de montrer qu'on est capable de modéliser un objet rigide par un objet peu déformable. Dans cet exemple, on compare la trajectoire d'une pyramide élastique lancée verticalement, et soumise à la gravité, avec la trajectoire d'un solide rigide. Pour cela, on compare la position du centre de gravité de la pyramide élastique avec celui de la pyramide rigide dont le mouvement est donné par les lois de la mécanique d'objets rigides.

Normalement quand on applique une force sur un objet déformable, une partie de l'énergie de cette force est utilisée pour se déformer et le reste pour se déplacer. Quand l'objet est rigide, cette énergie est totalement utilisée pour se déplacer. La pyramide est définie par un ensemble de particules (figure 6.9) liées par deux types de connecteurs: des connecteurs linéaires et des connecteurs angulaires. On considère que les connecteurs linéaires sont identiques, que leur coefficient de rigidité est λ_1 et que leur coefficient de viscosité est μ_1. On considère également que les connecteurs angulaires sont identiques, que leur coefficient de rigidité est λ_2 et que leur coefficient de viscosité est μ_2.

Le but consiste à trouver les valeurs des quatre paramètres précédents pour que la différence entre la trajectoire de la pyramide rigide et celle de la pyramide déformable soit inférieure à un certain seuil $\epsilon = 9e^{-4}$. Cela peut être obtenu pour des valeurs très grandes de ces paramètres mais cela peut ralentir énormément le temps d'exécution. Pour cela, on veut ajouter un autre critère concernant la vitesse d'exécution qui peut être contrôlé en agissant sur la tolérance en énergie mécanique (§ 3.4). Le critère d'erreur, dans ce cas, sera prioritaire à celui de la vitesse. Le

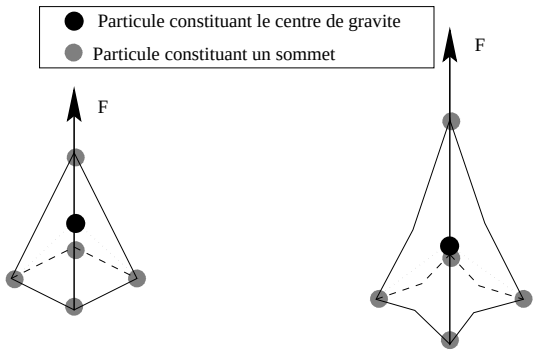


Figure 6.9: *La modélisation d'une pyramide déformable.*

critère de la vitesse ne sera pris en compte que, lorsque le critère de l'erreur est satisfait (figure 6.10).

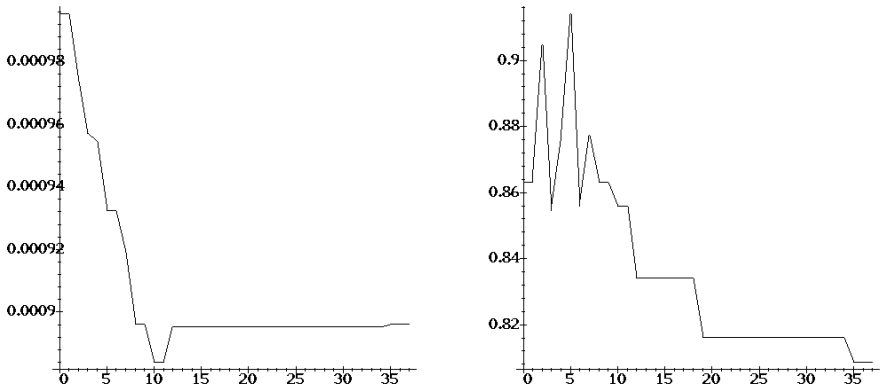


Figure 6.10: *A gauche on voit l'erreur en position et à droite on voit le temps d'exécution.*

On peut constater, d'après la figure 6.10, que jusqu'à la $7^{\text{ième}}$ génération, le critère était celui de l'erreur sur la position, tandis qu'à partir de la $8^{\text{ième}}$ génération, on se base aussi sur la vitesse d'exécution. Cela se traduit par une décroissance de l'erreur sur la position jusqu'à la septième génération de façon monotone, puis après cette génération, on ne dépasse plus le seuil d'erreur qui était fixée à 9.10^{-4}. Toutes les valeurs suivantes sont alors inférieures à ce seuil. En revanche, sur le graphe de droite, qui représente le temps d'exécution du programme, on peut voir que ses débuts sont très chaotiques, puisque le critère de vitesse d'exécution n'intervient pas jusque là. En revanche, à partir de la $8^{\text{ième}}$ génération, le temps décroît lui aussi de façon monotone.

Au bout de la $37^{\text{ième}}$ génération, on obtient une erreur égale à 0.00089 et un temps d'exécution égale à 0.8 secondes.

Construction d'une chaîne articulée: Une chaîne articulée est faite par un ensemble de corps rigides dont les points de contact sont liés entre eux par des ressorts linéaires très rigides (figure 6.11).

Le but de ce test est de trouver les paramètres λ et μ des ces ressorts linéaires afin que l'écart entre les différents corps rigides soit presque nul tout en optimisant le temps d'exécution.

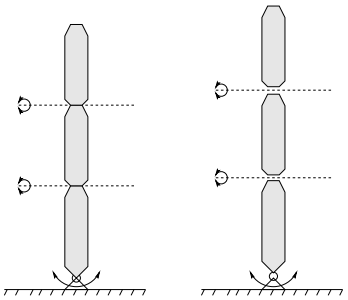


Figure 6.11: *Une chaîne articulée.*

L'identification est faite deux fois, une fois par un opérateur humain et une autre fois par l'algorithme génétique.

L'opérateur humain a mis 8 heures pour trouver les valeurs suivantes: $\lambda = 200$, $\mu = 100.5$ et la tolérance en énergie $\epsilon_e = 0.0001$. L'écart maximal était d'environ $2.0e^{-7}$ et le temps d'exécution était de 157 secondes.

L'algorithme génétique (figures 6.12 et 6.13) a mis une nuit pour trouver les résultats suivants: $\lambda = 298.863$, $\mu = 50.153$ et la tolérance en énergie $\epsilon_e = 0.00004$. L'écart maximal était d'environ $4.05e^{-8}$ et le temps d'exécution était de 35 secondes pour exécuter la même tâche.

On remarque que bien que l'algorithme génétique donne une valeur plus petite pour la variation de l'énergie mécanique, le temps d'exécution était paradoxalement plus petit. Cela peut être expliqué par le fait que quand la résolution de l'équation différentielle commence mal, à cause d'une grande tolérance, elle aura plus de mal à compenser par la suite.

Traitement de deux critères dépendants

Un avantage majeur du simulateur dynamique est la possibilité de prendre, implicitement, en compte la dynamique du robot. Le but de ce test est de trouver les paramètres d'un contrôleur PD (proportionnel et dérivé), appliqué sur l'organe final d'une chaîne articulée, pour que cet organe arrive le plus vite possible (figure 6.14) à une position donnée à vitesse nulle. La force générée par un PD est:

$$F = -\lambda x - \mu v$$

où x est la distance entre la position actuelle de l'organe et la position finale, et v est la vitesse de cet organe.

Bien que la force de contrôle soit appliquée sur l'organe final, les positions, les vitesses et les accélérations des articulations passives (§ 5.2.2) nous donnent automa-

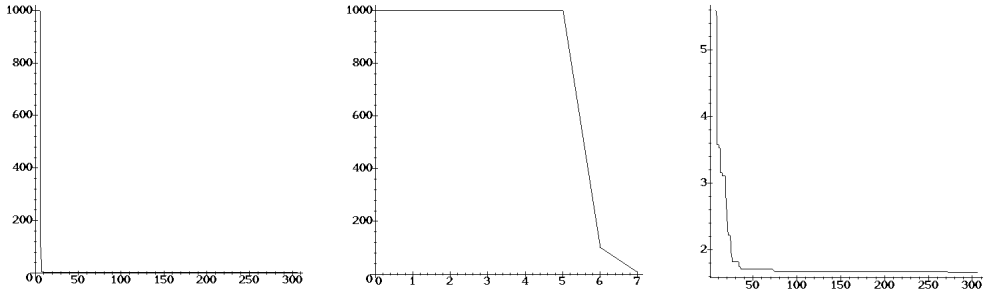


Figure 6.12: *La fonction d'évaluation sur l'ensemble des relations des meilleurs individus.*

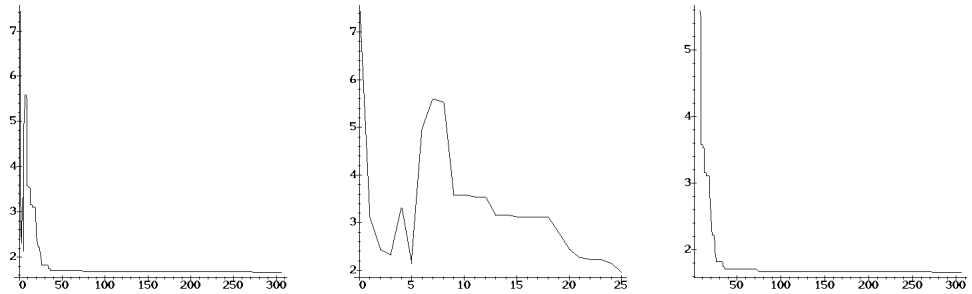


Figure 6.13: *Le temps d'exécution du meilleur individu pour chaque génération.*

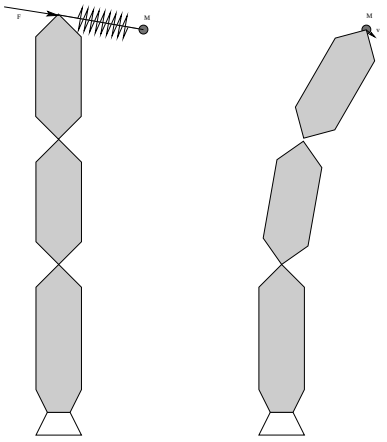


Figure 6.14: *Le doigt de la main de Salisbury, soumis à la force $\vec{F}$, doit atteindre le point M avec une vitesse $\vec{V}$ nulle.*

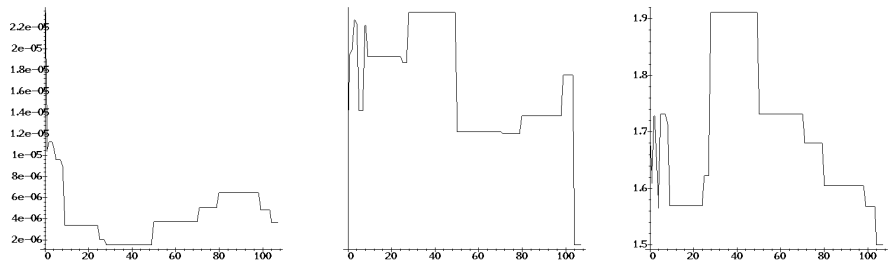


Figure 6.15: *De droite à gauche, la vitesse de l'organe final, sa distance et la vitesse d'exécution*

tiquement la loi de contrôle à appliquer sur ces articulations pour obtenir le même comportement.

Les trois critères qu'on cherche à satisfaire ne sont pas indépendants, car on cherche une vitesse nulle à la position finale et non pas n'importe où. L'erreur à atteindre est de 10^{-8} pour la distance et de 10^{-5} pour la vitesse. Donc la fonction de l'évaluation est une combinaison linéaire entre l'erreur en vitesse et celle en position.

La figure 6.15 montre l'évolution simultanée de la vitesse de l'organe final, sa distance, et la vitesse d'exécution.

On remarque qu'à partir de la $50^{\text{ième}}$ génération, des solutions qui satisfont les deux critères de la vitesse et de la position apparaissent : la vitesse d'exécution est donc prise en compte à partir de ce moment et on remarque qu'elle commence à descendre.

La figure 6.16 montre l'évolution de l'erreur globale en fonction du nombre de générations. On remarque que le résultat voulu est obtenu au bout de 100 générations où chaque génération contient 10 individus. Le résultat était $\lambda = 457.5734$, $\mu = 10.33436$, la distance du but $= 8.125e{-}9$, la vitesse de l'organe final à côté du but $= 3.7e{-}6$ et le temps nécessaire pour atteindre le but était de 1.5 secondes d'exécution

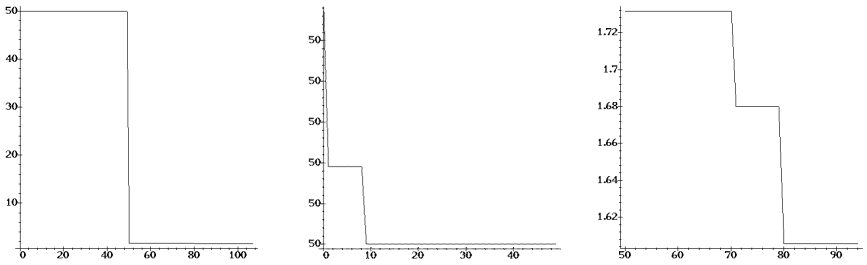


Figure 6.16: *L'évolution de l'erreur pendant la recherche par l'algorithme génétique.*

6.4 Conclusion

Le simulateur dynamique représente un objet par un modèle discrétisé et paramétré. La discrétisation structurelle de l'objet en plusieurs particules nécessite une distribution de la masse de cet objet entre ces particules tout en respectant ses propriétés d'inertie. Pour cela nous avons développé un algorithme qui consiste à diviser l'objet en plusieurs tétraèdres puis à remplacer chaque tétraèdre par cinq particules. Quatre particules aux sommets, dont les masses sont $\frac{1}{20}$ de la masse totale du tétraèdre, et une particule au centre d'inertie, dont la masse est $\frac{16}{20}$ de la masse totale. On a développé également des méthodes approximatives pour identifier les propriétés d'inertie en minimisant davantage le nombre de particules. Cette approche ne respecte pas seulement les propriétés d'inertie au niveau global, mais aussi au niveau local afin qu'elles restent respectées quand l'objet se déforme.

Les paramètres qui définissent les propriétés physiques de l'objet (élasticité, plasticité, etc...) ne peuvent pas être déterminés par une méthode numérique. En fait le modèle dynamique ne contient qu'un petit nombre de particules par rapport à la réalité. Pour cela ces paramètres ne correspondent pas forcément aux paramètres naturels. On cherche donc à trouver les valeurs de ces paramètres d'une manière expérimentale. Dans ce but on a utilisé et adapté la notion des algorithmes génétiques pour trouver d'une manière automatique les valeurs de ces paramètres.

Etant donné un comportement d'un objet représenté par un ensemble de contraintes (position et déformation en fonction du temps, position à un instant donné, déformation maximale, vitesse, accélération), les algorithmes génétiques nous permettent de trouver les paramètres qui induisent le même comportement.

Il nous reste à trouver un moyen pour apprendre le comportement d'un objet réel. Cela nécessite du matériel spécial pour pouvoir enregistrer la position, la vitesse, et les déformations d'un objet en fonction du temps.

Conclusion

Nous avons présenté, dans cette thèse, des modèles et des algorithmes conçus pour produire des simulations dynamiques efficaces et consistantes, dans le contexte de la Robotique d'intervention (c'est-à-dire, pour les tâches robotiques qui impliquent des contraintes fortes sur la nature de l'interaction entre des objets qui ne sont pas forcément rigides). Ces modèles et ces algorithmes ont été intégrés et implantés dans le système *Robot*Φ qui peut être potentiellement reconfiguré pour traiter une grande variété de tâches d'intervention, comme la manipulation dextre d'un objet par une main robotique, la manipulation d'un objet non rigide, la téléprogrammation du mouvement d'un véhicule tout-terrain, ou encore des tâches chirurgicales assistées par robot (par exemple, le positionnement d'un ligament artificiel dans la chirurgie du genou). Ce système est implanté en langage $C++$ sur des machines Silicon Graphics en utilisant *Geomview* [2] comme système de visualisation $3D$. L'interface utilisateur est un langage formel qui permet à définir de scènes, de forces, d'objets, de primitives, et de connecteurs. Il permet aussi de définir des actions qui permet de lier les différentes primitives ou de demander du système de le faire automatiquement. Les expressions arithmétiques sont du même type que les expressions en C. L'avantage d'une telle interface qu'elle n'est pas liée à une machine donnée et le système peut être compilé sur n'importe quel type de machine. Il est actuellement opérationnel sur Silicon Graphics, sur Sun, sur PC, et sur Macintosh. Ce langage est décrit en détail dans l'annexe B.

Le modèle utilisé est une généralisation de la notion du système masse/ressorts. Cette généralisation consiste à introduire d'autres types de primitives autre que les masses ponctuelles et à introduire d'autres types de connecteurs. Nous avons déjà introduit les objets rigides et nous voulons introduire également les chaînes articulées.

Quelque soit le type de la primitive, elle se représente dynamiquement par un ensemble de particules (masses ponctuelles) qui conservent ses propriétés d'inertie.

La forme géométrique d'un objet est représentée par un ensemble de polyèdres initialement convexes. Un polyèdre peut correspondre à une partie d'une primitive, à une primitive ou à plusieurs primitives.

Nous avons développé un algorithme adaptatif qui permet de passer automatiquement de la forme géométrique d'un polyèdre convexe à sa représentation dynamique. Le nombre de particules dépend seulement de la forme géométrique du polyèdre et de sa déformabilité.

[2] Geomview, est un programme interactif pour visualiser et manipuler des objets géométriques. Il est disponible et libre sur le ftp cite 'geom.umn.edu' (IP address 128.101.25.35).

Le mouvement et les déformations des objets sont caractérisés par un système d'équations différentielles dépendantes et discontinues (à cause des forces de collision et de contrôle). La résolution de ce système pose des problèmes numériques comme la divergence. Pour cela une approche adaptative basée sur la notion de l'énergie mécanique est développée. Cette approche permet d'estimer l'erreur commise, d'éviter la convergence numérique et d'améliorer la performance du système. La complexité du calcul du mouvement et des déformations d'un objet est $O(n)$, où n est le nombre de particules constituant l'objet.

Le fait que les objets sont déformables rend la détection du contact difficile et nécessite un temps d'exécution très important. Pour cela nous avons adapté l'algorithme de Gilbert, conçu initialement pour détecter le contact entre objets rigides, pour détecter et localiser le contact entre deux polyèdres déformables en mouvement. La complexité de cet algorithme est $O(n)$ ou n est le nombre total de sommets des polyèdres en question. Donc, la complexité totale du passage de la position d'un objet à l'instant t à sa position à l'instant $t + \tau$ est linéaire.

Les deux algorithmes, celui du mouvement et celui de la détection du contact sont paralèllisable et leur complexité parallèle est $O(1)$.

Trois types d'interactions sont prises en compte: la collision, le frottement et la viscosité du milieu. La force de la collision est calculée en utilisant un modèle masse/ressort non linéaire. L'avantage de ce modèle est que le coefficient de la restitution ne dépend que de la vitesse relative de deux objets en collision, ce qui est cohérent avec les résultats expérimentaux obtenus par les physiciens. Le frottement est modélisé par le modèle de Coulomb, et la viscosité est statistiquement représentée par une force qui dépend de la direction du mouvement, de la vitesse, de la forme géométrique et du milieu extérieur.

Le comportement d'un objet physique peut être contrôlé par l'intermédiaire des forces virtuelles. Ces forces ne sont pas le résultat d'un moteur ou d'un actionneur et elles peuvent être appliquées sur n'importe quel point de l'objet. La force virtuelle permet de reproduire des trajectoires causées par des actionneurs très complexes comme c'est le cas pour le mouvement passif du genou qui est produit par un manipulateur humain. Les forces virtuelles peuvent se propager à l'intérieur d'un robot pour donner automatiquement les configurations des actuateurs qui correspondent à la trajectoire reproduite.

Les algorithmes génétiques ont été adaptés pour identifier les paramètres physiques d'un objet (élasticité, plasticité, etc) ainsi que pour identifier les paramètres d'une force de contrôle capable de lui faire exécuter une tâche donnée.

Le système a été utilisé pour simuler le comportement dynamique du ligament croisé du genou au cours de son mouvement passif. Des données réelles ont été prises sur 4 ligaments de cochons. La différence entre les résultats obtenus par simulation et les données mesurées était d'environ 5%. Il est également utilisé pur simuler l'interaction main/objet pendant l'opération de la saisie. Une application industrielle avec la société Getris-Images a été lancée pour générer des effets spéciaux plus réalistes sur les images.

Perspective Jusqu'à maintenant, l'optimisation du système avait comme but de développer des algorithmes capables de rendre sa complexité linéaire $O(n)$ ou $k.n$.

La valeur de k n'a pas été optimisé pendant le travail de recherche. En fait on peut optimiser cette valeur en combinant plusieurs algorithmes (par exemple en utilisant des algorithmes différents pour détecter le contact entre les polyèdres rigides [69] et les polyèdres déformables).

La méthode d'intégration numérique n'est pas la seule source d'erreur. Cette erreur dépend aussi de la manière dont le programme est écrit (les opérations '-' et '+' par exemple ne sont pas associatives dans un langage de programmation) et elle dépend aussi de la façon dont l'équation du mouvement est trouvée (les méthodes analytiques par exemple peuvent minimiser cette erreur [1]). Le traitement de ces deux derniers types d'erreur est très important pour améliorer la performance du système.

Une version parallèle et câblée du système sera le sujet d'une coopération avec la société Getris-Images. Le but est de réaliser une version $3D$ de la carte DVE commercialisé par Getris. On espère que cette version tournera en temps réel afin de l'utiliser pour des applications robotiques qui exigent le temps réel.

La perspective à long terme de notre travail sur le ligament est de l'utiliser pour la planification chirurgicale. La précision de ce modèle le rend convenable pour une utilisation ultérieure afin d'optimiser le placement de l'LCA si les images MRI ou CT nous permettent d'identifier les surfaces des fibres.

Pour ce qui concerne la saisie par une main articulée, nous voulons, en coopération avec d'autres membres de l'équipe SHARP, comparer quantitativement les résultats obtenus par simulation avec la main réelle.

Acknowledgements

Ce travail est partiellement supporté par le CNES (Centre National des Etudes Spatiales) par l'intermédiaire du projet national RISP, et par la région Rhône-Alpes par l'intermédiaire du projet robotique SHARP (un projet IMAG/INRIA).

Bibliography

[1] B. Arnaldi, G. Dumont, and G. Hégron. Animation of physical systems from geometric, kinematic and dynamic models. In *Working Conference on Modeling in Computer Graphics*, Tokyo, Japan, april 1991. IFIP, Springer Verlag.

[2] Marc Atteia and Michel Pradel. *Eléments d'analyse numérique*. Cepadues-Editions.

[3] Smith BA, Livesay GA, and Woo SLY. Biology and biomechanics of the anterior cruciate ligament. *Clinics in Sports Medicine*, 12(4):637–670, 1993.

[4] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics, volume 23, Number 3*, july 1989.

[5] D. Baraff. Coping with friction for non-penetrating rigid body simulation. *Computer Graphics*, 1991.

[6] D. Baraff. Dynamic simulation of non-penetrating rigid bodies. Technical Report 92-1275, Cornell University, 1992.

[7] D. Baraff. Rigid body simulation. *SIGGRAPH*, 1992. Lecture notes.

[8] D. Baraff. Non-penetreating rigid body simulation. In *Eurographics*, 1993.

[9] D. Baraff and A. Witkin. Dynamic simulation of non-penetrating flexible bodies. *Computer Graphics, volume 26, Number 2*, july 1992.

[10] C. Bard. *Interaction Sensorio-Motrice en robotique: Application à la Prehension Automatisée pour une Main Articulée à Plusieurs Doigts*. PhD thesis, INPG: L'Institut National Polytechnique de Grenoble/France, 1994.

[11] Ferdinand P. Beer and jr E. Russell Johnston. *Vector Mechanics for Engineers: DYNAMICS*. McGraw-Hill Book Company, 1984.

[12] MZ Benbjaballah and a Shirazi-Adl. Biomechanics of the human knee joint in compression: reconstruction, mesh generation, and finite element analysis. *The knee*, 2(2):69–79, 1995.

[13] E. Bertin and J.M Chassery. Diagramme de voronoï 3d: construction et applications en imagerie 3d. In *AFCET 8ième Congrès de Reconnaissance de Formes et Intelligence Artificielle*, Lyon, France, Novembre 1991.

[14] Eric Bittar, Nicolas Tsingos, and Marie-Paule Gascuel. Automatic reconstruction of unstructured 3d data: Combining medial axis and implicit surfaces. In *Eurographics*, Maastrich, Pays Bas, sept 1995.

[15] L Blankevoort and R. huiskes. Ligament-bone interaction in a 3-dimensional model of the knee. *Biomechanics. Eng.*, 113:263–269, 1991.

[16] C. Cadoz and JL. Florens. Cordis-anima: a modeling and simulation system for sound and image synthesis. *Computer Music*, 1993.

[17] Stephane H. Carandall, Dean C. Karnopp, Edward F. Krtz, and David C. Pridmore-Brown. *Dynamics of mechanical and electronical systems.* Krieger publishing company, Malabar, Florida, 1968.

[18] Michel Carignan, Ying Yang, Nadia Magnenat Thalmann, and Daniel Thalmann. Dressing animated synthetic actors with complex deformable clothes. In *Computer Graphics*, July 1992.

[19] Thierry Carlier and Nicolas Szafran. Structure de données pour la représentation et manipulation des polyèdres. Technical Report RR 772-M, IMAG, Mars 1989.

[20] D. Casanova and C. Laugier. Generating environmenets for a physical simulation of robotic tasks. *International conference ORIA*, Decembre 1994.

[21] Benoit Chanclou, Annie Luciani, and Arash Habibi. Physical models of loose soils dynamically marked by a mobile object. In *IEEE/Computer Animation*, Geneva, Switzerland, 1996.

[22] M. Cherif. *Planification de mouvements pour un robot mobile autonome tout terrain : une approche par utilisation des modèles physiques.* PhD thesis, INPG: L'Institut National Polytechnique de Grenoble/France, 1995.

[23] Rémi Cozot and Bruno Arnaldi. Dream: modèle de matière pour l'animation. In *Association Française de l'Informatique Graphique FFIG*, Marseille, France, 95.

[24] Rémis Cozot and Bruno Arnaldi. A language for multibody systems modelling and simulation. In 7^{th} *European Simulation Symposium ESS*, Nurembreg, Germany, octobre 95.

[25] Umberto Cugini, Paolo Denti, Massimo Ippolito, Caterina Rizzi, and Michele Ursino. A system to simulate non-rigid materials behaviour during handling processes. In *2nd Japan-France Congress on Mechatronics*, 1994.

[26] P. Dario, C. Paggetti, T. Clucci, B. Allotta, M. Marcacci, M. Fadda, S. Martelli, and D. Caramella. A system for computer-assisted articular surgery. *Computer Aided Surgery*, 1, october 1995.

[27] H. Delingette, Gérard Subsol, Stéphane Cotin, and Jérôme Pignon. *In Richard A. Robb, editor, Visualization in Biomedical Computing*, chapter A Craniofacial Surgery Simulation Testbed. VBC'94, Rochester (Minnesota) (USA), october 1994.

[28] H. Delingette, Y. Watanabe, and Y. Suenaga. *In N. Magnenat-Thalmann and D. Thalamann, editors, Models and Techniques in Computer Animation*, chapter Simplex based animation. Computer Animation, Springer Verlag, Geneva, Switzerland, june 1993.

[29] Hervé Delingette, Gérard Subsol, Stéphane Cotin, and Jérome Pignon. A craniofacial surgery simulation testbed. research report RR-2199, INRIA, February 1994.

[30] J. Dorlot, J. Batlon, and J. Masounve. *DES MATRIAUX*. Ecole Polytechnique de Montereal, 1986.

[31] Jean-Luc Lutton Ernesto Bonomi. Le recuit simulé. *Pour la science*, (129):68–77, juillet 1988. Derive de methodes de physiques statistique, le recuit simule resout efficacement des problemes d'optimisation. Il permet notamment de determiner l'architecture de grands reseaux teleinformatique.

[32] Fabien Garat. Identification de paramètres physiques pour un simulateur dynamique appliqué à la robotique. Master's thesis, INPG, Grenoble, France, juin 1996.

[33] Alejandron Garcia-Alonso, Nicolas Serrano, and Juan Flaquer. Solving the collision detection problem. *IEEE/Computer Graphics and Applications*, 1994.

[34] Jean-Dominique Gascuel and Marie-Paule Gascuel. Displacement constraints for interactive modeling and animation of articulated structures. *The visual Computer*, 1994.

[35] M. Gascuel, A. Verroust, and C. Puech. A modelling system for complex deformable bodies suited to animation and collision processing. *Visualisation and computer animation*, 1991.

[36] D. Gay and J. Gambelin. *Une approche simple du calcul des structures par la méthode des éléments finis*. Hermès, 1989.

[37] P.L George. Géneéation automatique de maillages: application aux méthodes d'élement finis. Technical report.

[38] E.G. Gilbert and S.M. Hong. A new algorithme for detecting the collision of moving objects. *IEEE/International Conference on Robotics and Automation "ICRA"*, 1989.

[39] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. A fast procedure for computing the distance between objects in three-dimentional space. *IEEE/International Conference on Robotics and Automation "ICRA"*, 1988.

[40] David E. Goldberg. *Algorithmes génétiques. Exploration, optimisation et apprentissage automatique.* Addison-Wesley, 1996.

[41] W. Goldsmith. *Impact: The theory and physical behaviour of colliding solids.* London: Edward Arnold Ltd, 1960.

[42] Herbert Goldstein. *Classical Mechanics.* Addison-Wesley, 1980.

[43] Jean-Paul Gouret. Simulation of object and human skin deformations in a grasping task. *Computer Graphics*, 1989.

[44] Jean-Paul Gouret, Nadia Magnenat Thalmann, and Daniel Thalmann. Simulation of object and human skin deformations in a grasping task. In *Computer Graphics*, July 1989.

[45] J.-P. Gourret. *Modélisation d'images fixes et animées.* Masson, 1994.

[46] Thiery Guirard-Marigny, Nicolas Tsingos, Ali Adjoudani, Christian Benoit, and Marie-Paule Gascuel. 3d models of the lips for realistic speech animation. In *IEEE/Computer Animation*, Geneva, Switzerland, 1996.

[47] K. H Hunt and F. R. E. Crossley. Coefficient of restitution interpreted as damping in vibroimpact. *ACME Journal of Applied Mechanics*, 1975.

[48] David Crochemore Jean Louchet, Xavier Provot. Evolutionary identification of cloth animation models. In *Computer Animation and Simulation.* Proceedings of the Eurographics Workshop, 1995.

[49] S. Jimenez, A. Luciani, O. Raoult J.L. Florens, and C. Cadoz. Modèles comportementaux vers une approche instrumentale de la synthèse d'image. Technical report, ACROE/INPG, 1989. In french.

[50] S. Jimenez, A. Luciani, and C. Laugier. Teleprogramming the motion of a planetary robot using physical models and dynamic simulation tools. *IEEE International workshop on intelligent robots and systems*, july 1990.

[51] S. Jimenez, A. Luciani, and C. Laugier. Physical modeling as an help for planning the motion of a land vehicle. *IEEE/RSJ International workshop on intelligent robots and systems*, November 1991.

[52] S. Jimenez, A. Luciani, and C. Laugier. Predicting the dynamic behaviour of a planetary vehicle using physical models. *IEEE/RSJ International workshop on intelligent robots and systems*, July 1993.

[53] D. W. Johnson. The optimisation of robot motion in the presence of obstacles. *PHD, University of Michigan*, 1987.

[54] A. Joukhadar. Energy based adaptive time step and inertia-matrix based adaptive discretization for fast converging dynamic simulation. *International workshop on Visualisation and Mathematics*, May 1995.

[55] A. Joukhadar. *robotϕ*: Dynamic modeling system for robotics applications. research report RR-2543, INRIA, May 1995.

[56] A. Joukhadar, C. Bard, and C. Laugier. Combining geometric and physical models, the case of a dextrous hand. *IEEE/International Conference on Intelligent Robots and Systems "IROS"*, Septembre 1994.

[57] A. Joukhadar, C. Bard, and C. Laugier. Planning dextrous operations using physical models. *IEEE/International Conference on Robotics and Automation "ICRA"*, May 1994.

[58] A. Joukhadar and Ch. Bard. Modélisation physique pour la planification de saisie avec une main articulée. research report RR-2247, INRIA, Avril 1994. In french.

[59] A. Joukhadar and C. Laugier. Dynamic modeling of rigid and deformable objects for robotic task: motions, deformations, and collisions. *International conference ORIA*, Decembre 1994.

[60] A. Joukhadar and C. Laugier. *robotϕ* : A physical modeling system for robotic applications. *Japan-France congress on MECHATRONICS*, November 1994.

[61] A. Joukhadar and C. Laugier. Dynamic modeling and its robotics applications. *IEEE/International Conference on Robotics and Automation "ICRA"*, May 1995.

[62] A. Joukhadar and C. Laugier. Fast dynamic simulation of rigid and deformable objects. *IEEE/International Conference on Intelligent Robots and Systems "IROS"*, August 1995.

[63] A. Joukhadar and C. Laugier. Dynamic simulation: Model, basic algorithms, and optimization. In *Workshop on the Algortihmic Fondations of Robotics*, Toulouse, France, July 1996.

[64] A. Joukhadar, A. Wabbi, and Ch. Laugier. Fast contact localisation between deformable polyhedra in motion. *IEEE/Computer Animation*, June 1996.

[65] Alexis Lamouret and Marie-Paule Gascuel. An approach for guiding colliding physically-based models. In *Third Workshop on Animation & simulation, Barcelona*, 1993.

[66] C. Laugier, C. Bard, M. Cherif, and A. Joukhadar. Solving complex motion planning problems by combining geometric and physical models: the case of a rover and of a dextrous hand. *Workshop on the Algortihmic Fondations of Robotics (WAFR)*, February 1994.

[67] C. Laugier, C. Bard, M. Cherif, and A. Joukhadar. *In K. Goldberg, D. Halperin, J-C. Latombe, and R. Wilson (Eds)Algorithmic Foundation of Robotics*, chapter Solving complex motion planning problems by combining geometric and physical models: the case of a rover and of a dextrous hand. Eds, USA, 1995.

[68] Paul U. Lee, Diego C. Ruspini, and Oussama Khatib. Dynamic simulation of interactive robotic environment. In *IEEE/International Confernece on Robotics and Automation "ICRA"*, San Diego, USA, May 1994.

[69] Ming C. Lin and John F. Canny. A fast algorithm for incremental distance calculation. *IEEE/International Conference on Robotics and Automation "ICRA"*, April 1991.

[70] DA Loch, Z Luo, J Lewis, and N Stewart. A theoretical model of the knee and acl: theory and experimental verification. *Biomechanics*, 25(1):81–90, 1992.

[71] Jean-Christophe Lombardo and Claude Puech. Modélisation d'objets déformables avec un système de particules orientées. *Revue Internationale de CFAO et d'informatique graphique*, Paris, 1995.

[72] A. Luciani and al. An unified view of multiple behaviour, flexibility, plasticity and fractures: balls, bubbles and agglomerates. In *IFIP WG 5.10 on Modeling in Computer Graphics*, pages 55–74. Springer Verlag, 1991.

[73] Duane W. Marhefka and Davide E. Orin. Simulation of contact using a nonlinear damping model. In *IEEE/International Conference on Robotics and Automation "ICRA"*, Minneapolis, Minnesota, April 1996.

[74] S. Martelli, A. Joukhadar, S. Lvallee, G. Champleboux, and M. Marcacci. Acl mathematical model for implant simulation in animal. *To be presented in the 7th Congress of the European Society of Sports Traumatology, Knee Surgery and Arthroscopy (ESSKA)*, May 1996.

[75] S. Martelli, A. Joukhadar, S. Zaffagnini, M. Marcacci, S. Lavallee, and G. Champleboux. A fiber-based acl model for geometrical and mechanical simulations. research report RR-2924, INRIA, July 1996.

[76] M.T. Mason and J.K. Salisbury. *Robot hands and the mechanics of manipulation.* Artificial Intelligence. MIT Press, 1985.

[77] J. Max. Méthodes et techniques de traitement du signal et applications aux mesures physiques. *Masson*, 1989.

[78] D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constraints. *Computer Graphics, volume 21, Number 2*, july 1992.

[79] Michel Minoux. *Programmation mathématique - Théorie et algorithme.* Coll. Technique et Scientifique des Telecommunications, Dunod, 1983.

[80] Brian Mirtich and John F. Canny. Impulse-based, real time dynamic simulation. *Workshop on the Algortihmic Fondations of Robotics (WAFR)*, February 1994.

[81] M. Moore and J. Wilhems. Collision detection and response for computer animation. In *Computer Graphics*, Augest 1988.

[82] Jamel Nouiri, Claude Cadoz, and Annie Luciani. The physical modelling of complex physical structures: The mechanical clockwork, motion, imag and sound. In *IEEE/Computer Animation*, Geneva, Switzerland, 1996.

[83] William H. Press, Saul A. Teukolsky, and Brian P. Flannery. *Numerical Recipes in C.* Cambridge Unversity Press, 1992.

[84] E. Promayon, P. Baconnier, and C. Puech. Physically-based deformation constrained in displacements and volume. In *EUROGRAPHICS*, Futuroscope Poitiers - France, August/Août 1996.

[85] L.F Shampine and Gordon M.K. *Computer Solution of ordinary differential equations: The initiale value problem.* Freeman, 1975.

[86] M. Spong and M. Vidyasagar. *Robot Dynamics and Control.* John Wiley & Sons.

[87] Richard Szeliski and Stéphane Lavallée. Matching 3-d anatomical surfaces with non-rigid deformations using octree-splines. In *Workshop on BIA*, 1994.

[88] Nicolas Szilas. *Apprentissage dans les réseaux réccurents pour la modélisation mécanique et étude de leurs interactions avec l'environnement.* PhD thesis, INPG, 1994.

[89] K. Komoriya T. Kotoku and K. Tanie. A force display system for virtual environments and its evaluation. *IEEE/International Workshop on Robot and Human Communication (RoMan'92)*, September 1992.

[90] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *Computer Graphics, volume 21, Number 4*, july 1987.

[91] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformables models. *Computer Graphics*, august 1988.

[92] Nicolas Tsingos, Eric Bittar, and Marie-Paule Gascuel. Semi-automatic reconstruction of implicit surfaces for medical applications. In *Computer Graphics International*, United Kingdom, june 1995.

[93] Manuel Da Vinha. Modélisation automatique d'objets physiques. Master's thesis, INPG, Grenoble, France, juin 1995.

[94] Pascal Volino, Martin Courchesne, and Nadia Magnenat Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. *Computer Graphics Proceedings*, 1995.

[95] Pascal Volino and Nadia Magnenat Thalmann. Collision and self-collision detection: Efficient and robust solutions for highly deformable surfaces. *Eurographics Workshops in Maastricht, The Netherland*, 1995.

[96] Y. Wang and M. T. Mason. Two-dimensional rigid-body collisions with friction. *ASME Journal of Applied Mechanics*, september 1992.

[97] J Wismans, F Velpaus, F Janssen, J Huson, and P Struben. A three-dimensional mathematical model of the knee joint. *Biomechanics*, 13:677–685, 1980.

[98] Andrew Witkin and William Welch. Fast animation and control of nonrigid structures. *Computer Graphics*, 1990.

[99] SL Woo, GA Johnson, and BA Smith. Mathematical modeling of ligaments and tendons. *Trans. of ASME*, pages 468–473, 1993.



Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399