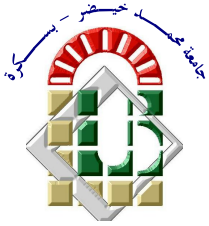
الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Khider – Biskra
Faculté des Sciences Exactes et des
Sciences de la Nature et de la vie
Département d'informatique
Réf :……



جامعة محمد خيضر بسكرة
كلية العلوم الدقيقة و علوم الطبيعة و الحياة
قسم الإعلام الآلي
المرجع: ……

Mémoire en vue de l'obtention
du diplôme de

Magister en informatique

Option : Synthèse d'image et vie artificielle

Intitulé :

Gestion de collision en animation de vêtements

Présenté par :

ZOUIOUECHE Amina

Soutenu publiquement le :

Devant le jury composé de :

Dr. BABAHENINI Med Chaouki	**Maître de Conférences 'A'**	**Président**	**Université de Biskra**
Pr. DJEDI NourEddine	**Professeur**	**Rapporteur**	**Université de Biskra**
Dr. BAARIR Zineddine	**Maître de Conférences 'A'**	**Examinateur**	**Université de Biskra**
Dr. CHERIF Foudil	**Maître de Conférences 'A'**	**Examinateur**	**Université de Biskra**
Dr. MELEKEMI Kamel Eddine	**Maître de Conférences 'A'**	**Examinateur**	**Université de Biskra**

Remerciements

Je tiens d’abord à remercier mon encadreur qui m’a soutenu tout au long de la rédaction de ce mémoire, pour son écoute et ses conseils pertinents, et surtout pour sa patience.

Je remercie également l’ensemble des membres du jury pour avoir accepter de consacrer leur temps à examiner ce travail malgré leurs nombreuses responsabilités et occupations, je leur suis reconnaissante pour l’attention qu’ils ont portée à mon travail.

Je tiens aussi à remercier mon mari Mr. ZERNADJI Tarek pour toute l’aide qu’il m’a apporté, pour son soutient inconditionnel et sa présence continue.

J’aimerais également remercier mes parents et ma famille pour leurs encouragements, et motivations.

Enfin je remercie tous ceux qui m’ont aidé de près ou de loin pour la réalisation de ce travail.

Table des matières

Table des Figures

Résumé

Avec la croissance de l'industrie d'animation, le besoin d'une simulation de tissu efficace et visuellement attrayante devient immense. Mais la nature souple du tissu rend le problème très complexe, surtout quand il s'agit de la gestion des collisions du tissu avec son environnement.

Notre travail s'inscrit dans le cadre de la simulation comportementale. Son principal objectif est de proposer une solution au problème de la gestion des collisions pour la simulation des tissus en temps réel. Le modèle exploite la méthode accélératrice des volumes englobants et tire profit des techniques des niveaux de détails pour leurs propriétés d'accélération et ce dans le but de proposer un compromis entre rapidité de calcul et précision de la détection.

Abstract

With the growth of the animation industry, the need for efficient and visually appealing cloth simulation becomes huge. Meanwhile, the flexible nature of the fabric makes the problem very complex, especially when it comes to collision handling of the fabric with its environment.

Our work is a part in behavioral simulation. Its main objective is to propose a solution to the problem of collision handling for real time cloth simulation. The model applies the method of bounding volumes hierarchy and takes advantage of levels of detail techniques for their acceleration properties and this in order to find a compromise between computational speed and detection accuracy.

ملخص

مع نمو صناعة الرسوم المتحركة، أصبحت الحاجة إلى محاكاة القماش و الألبسة كفاءة وجمالا وجاذبية ملحة جدا. وفي الوقت نفسه، و نظرا للطبيعة المرنة الخاصة بالنسيج تعد هذه المهمة من أكثر المشاكل تعقيدا، خصوصا عندما يتعلق الأمر بالتعامل مع تصادم النسيج مع محيطه.

يشكل عملنا هذا جزءا من عملية محاكاة السلوك. و يكمن الهدف الرئيسي من هذا العمل إلى اقتراح حل لمشكلة التعامل مع مشكلة التصادم لمحاكاة القماش في وقت آني. يستفيد هذا النموذج من عملية تسريع الحسابات عن طريق استعمال الأحجام المحيطة إضافة إلى استغلاله من تقنيات مستويات التفصيل ، وهذا من أجل إيجاد حل وسط بين سرعة الحساب و دقة كشف التصادم.

Introduction générale

La croissance de l'industrie de l'animation numérique a permis aux applications de la réalité virtuelle de s'imposer dans notre quotidien, où l'on trouve les simulations virtuelles dans l'industrie du cinéma, les jeux, la médecine, la robotique, l'architecture, etc. Cette technologie temps réel, permet de simuler sur ordinateur un environnement virtuel 3D, dans lequel on peut évoluer, et donner l'impression d'une immersion dans le monde réel. En effet, ce, monde virtuel cherche à imiter le monde réel, aussi bien au niveau de ces propriétés statiques (mêmes apparences des objets qui le composent) qu'au niveau de ces propriétés dynamiques (mêmes comportements de ces objets). Ceci mène à l'adoption par la réalité virtuelle des lois de la physique et de la dynamique qui manipulent le monde réel.

Ainsi, des personnages virtuels évoluent dans cet environnement virtuel pour lesquels l'obtention d'un niveau de réalisme acceptable passe nécessairement par l'habillage de ces personnages par des vêtements.

Les travaux de recherche réalisés dans le domaine de la simulation de tissus et de vêtements sont apparus depuis plus d'une vingtaine d'années, ces travaux ont trois objectifs principaux:

- La quête du réalisme;
- L'optimisation du temps de calcul;
- L'interactivité.

L'interactivité se traduit par la faculté de rendre le vêtement prédictif indépendant de l'infographe. Ce qui est un but dur à atteindre vu que nous devons également respecter le réalisme et le temps de calcul.

Le temps d'exécution d'une simulation est une donnée critique pour ce genre d'applications, il est donc important d'avoir des résultats qui soient visuellement plausibles, sans que le temps nécessaire à l'obtention d'une image ne devienne prohibitif.

Malgré tous les travaux réalisés dans ce domaine, l'animation de tissus demeure très complexe. D'une technique de simulation à l'autre, la qualité des résultats obtenus, le temps de

calcul et la facilité d'utilisation varient énormément. Étant donné que les tissus et les vêtements se retrouvent pratiquement partout dans notre vie quotidienne, l'œil peut difficilement être trompé lorsqu'il est question de réalisme. Même avec toutes les avancées, il demeure relativement couteux de simuler des tissus réalistes. Dans ce cas c'est la recherche de la justesse physique de la simulation qui prime. Cette dernière impose aux chercheurs de s'intéresser aux interactions du tissu avec son environnement et ses interactions avec lui-même. Ceci met en valeur la gestion des collisions qui à comme rôle essentiel de détecter les interactions puis les traiter, ou dans d'autres applications prévoir les interactions pour pouvoir les éviter. 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 des déformations du tissu. Pour cela, de nombreuses méthodes accélératrices de la détection de collision ont été développées, allant du partitionnement de l'espace aux volumes englobants en exploitant la cohérence temporelle et spatiale, les stratégies exploitant les capacités de calcul des cartes graphiques et des GPUs, les méthodes stochastiques, etc.

Notre travail s'inscrit dans le cadre de la simulation comportementale. Son principal objectif est de proposer une solution pour la simulation des tissus en temps réel. Plus précisément, le développement d'un ensemble de méthodes pour résoudre les différents problèmes que pose l'animation temps réel des tissus, c'est à dire la simulation des propriétés mécaniques des tissus afin de reproduire de la façon la plus fidèle et la plus précise le comportement des tissus ainsi que la gestion des collisions, en tirant profit des techniques des niveaux de détails pour leurs propriétés d'accélération, afin de trouver un compromis entre rapidité de calcul et précision de la détection.

Le modèle proposé gère les collisions dans une simulation de tissu. En utilisant les volumes englobants et la multi résolution pour accélérer le processus de détection de collisions.

Le présent mémoire est organisé en quatre chapitres :

- Dans le premier chapitre, nous présentons le concept de collision et tout ce qui se cache derrière ce terme. Nous commençons par une brève présentation de la détection de collision qui d'une part garantit le réalisme de la simulation et d'autre part augmente sa complexité. Pour gérer cette complexité, plusieurs notions ont été abordées, comme la cohérence spatiale et la cohérence spatio-temporelle. Ensuite,

nous présentons les différentes méthodes accélératrices qui visent à réduire la complexité de la détection de collisions ainsi que son traitement. Nous relatons, également un bref aperçu sur un cas spécial de la collision, qui est l'interaction d'un objet avec lui-même, appelée encore auto-collision.

- Le deuxième chapitre est destiné à la simulation de vêtements, en expliquant les différents types et les méthodes utilisées, ensuite nous abordons la gestion de collisions dans la simulation des vêtements, en commençant par expliquer la méthode basique de la détection de collisions, suivie des différentes techniques accélératrices visant à améliorer la performance et la qualité de la simulation. Nous abordons ensuite le traitement de collisions et ces différentes méthodes dédiées à la simulation des tissus, puis nous donnerons une idée générale sur les auto-collisions ainsi que les collisions continues.
- Dans le troisième chapitre, nous exposons notre travail, qui consiste à gérer les collisions en utilisant la méthode d'accélération de la détection basée sur la hiérarchie des volumes englobants associée à la multi résolution. Nous commencerons par présenter la méthode de la hiérarchie des volumes englobants que nous avons choisie, en expliquant les détails de notre approche qui consiste à créer la hiérarchie des sphères englobantes et définir les différentes fonctions qui lui sont associées. Ensuite, nous expliquons comment nous avons introduit la multi résolution avec la hiérarchie des sphères englobantes pour accélérer la détection des collisions. Nous finissons par décrire la méthode de traitement de collision choisie.
- Le dernier chapitre est destiné à l'implémentation et les résultats. Ainsi, nous décrivons l'environnement de travail, les algorithmes utilisés, ainsi que l'analyse des résultats obtenus et l'évaluation du modèle.

Enfin, ce mémoire est achevé par une conclusion générale et des perspectives.

Chapitre I - Gestion de collisions :

Etat de l'art

1. Introduction

Un simulateur dynamique est, en général, un programme qui fait un nombre d'opérations très important [2]. Il simule le mouvement, la déformation et les interactions entre les objets. Il nécessite une interactivité complète et des taux de rafraichissement élevés pour permettre un rendu réaliste temps réel. Pour garantir cet effet de réalisme, un simulateur doit représenter le mouvement des objets composant la scène en fonction des forces extérieures, et l'effet de ces forces sur la forme géométrique des objets (déformation). Une étape importante, après avoir étudié les forces affectant le mouvement des objets, consiste à prendre en compte les interactions entre ces objets [1]. Ces interactions sont le vrai garant du réalisme de la simulation puisque c'est par échange d'énergie lors des contacts que le comportement des objets va évoluer. Une simulation sans interactions ressemble à un monde d'objets isolés. Ces interactions consistent principalement à détecter les collisions entre différents objets, localiser les régions de contacts et calculer les forces auxquelles sont soumis ces derniers.

Dans ce chapitre nous présentons ces interactions et tous les concepts sous-jacents nous commenceront par une brève présentation de la détection de collisions qui d'une part garantit le réalisme de la simulation et d'autre part augmente sa complexité. Pour gérer cette complexité, plusieurs notions ont été abordées, comme la cohérence spatiale et la cohérence spatio-temporelle. Nous allons d'abord présenter la gestion de la complexité puis discuter ses différentes notions ainsi que les différentes méthodes accélératrices qui visent à réduire la complexité de la détection de collision. Pour finir, nous allons présenter le traitement ou la réaction à cette collision, sa définition et les différentes méthodes permettant sa prise en charge. Nous donnons aussi pour clôturer ce chapitre un bref aperçu sur un cas spécial de la collision, qui est l'interaction de l'objet avec lui-même, ou les auto-collisions.

2. Présentation de la gestion des collisions

Le domaine de la simulation sur ordinateur utilise de plus en plus les techniques de simulation physique pour augmenter le réalisme des différentes scènes simulées. Pour ce faire, il est nécessaire de prendre en considération les interactions entre les entités composant ces scènes. Ces interactions doivent être reproduites par une gestion rigoureuse des collisions entre l'objet et l'environnement auquel il appartient ainsi que les collisions de l'objet avec lui-même. Cependant, il est utile de mettre en œuvre un module spécial

dont le but est d'identifier les instants et/ou les localisations de contacts ou d'interactions entre les entités simulées [4, 05]. Ce module s'identifie par la gestion des collisions qui se déroulent en deux étapes. La première est de détecter les interactions, c'est à dire où et quand deux objets quelconques de la scène entrent en contact, c'est ce que nous appelons détection de collisions. Cette détection génère un coût en exécution très important, plus encore lorsque les objets ont une forme variable [6]. Une fois les parties des objets entrants en collision identifiées, la deuxième partie consiste à déterminer la réaction de ces objets à cette collision, le comportement de ces derniers dépend des forces de contact entre eux et des frottements éventuels s'opposant aux glissements.

Dans ce qui suit, nous allons aborder en détail la détection de collisions ainsi que les techniques permettant son traitement.

2.1 Détection de collisions

Le terme détection de collision est souvent rencontré avec des dénominations diverses selon le type d'application concernée. On parle souvent de détection d'intersection, d'interférence, de recouvrement ou d'interpénétration lorsqu'il s'agit de détecter si deux objets virtuels s'interpénètrent ou pas. On parle aussi de détection de contacts lorsqu'il s'agit de définir les localisations où les objets virtuels sont en contact, à cette appellation est associé souvent la notion sous-jacente de topologie de contact qui fait que l'on ne s'arrête pas à une détection binaire. Dans certaines applications, la détection de contact sous-entend aussi la possibilité de déterminer le premier instant de collision. Enfin, on parle aussi de détection de proximité lorsqu'il s'agit d'algorithmique liée à la détermination de seuils de distances ou la distance minimale séparant une paire d'objets virtuels [5].

Une collision entre deux objets est détectée s'il existe un instant où les deux objets ont une intersection non vide [3]. Ainsi, détecter une collision revient à trouver d'une façon efficace le contact géométrique entre deux surfaces complexes [7].
Selon plusieurs études, le temps nécessaire à la détection d'une collision, dans le contexte de la simulation physique, peut représenter jusqu'à 99% du temps de calcul total. On constate donc qu'il faut absolument avoir recours à des algorithmes d'accélération de la détection de collisions qui, pour être justifiés, doivent être simples et rapides [3]. Avant d'aborder ces méthodes accélératrices, nous présenterons d'abord la complexité géométrique due au module chargé de la détection et de la gestion de collisions.

2.2 Gestion de la complexité géométrique

La gestion de la complexité géométrique vise à réduire la complexité du processus de détection de collisions dans le cadre de la simulation.
La détection de collisions entre plusieurs objets composant une scène virtuelle sera représentée par un algorithme naïf de complexité $O(n^2)$:

Algorithme : DCPlusieursObjets[2]

Données : tableau d'objet *objet* [ ], et le nombre d'objets *m*

1. Initialisation : i =1 ; j =1 ;
2. **tant que** $i \leq m$ **faire**
3. $j = i + 1$;
4. **tant que** $j \leq m$ **faire**
5. DCDeuxObjets (*objet* [*i*], *objet* [*j*]) ;
6. $j = j + 1$;
7. **fin**
8. $i = i + 1$;
9. Fin

Nous remarquons bien que la combinatoire est impressionnante, notamment si on prend en compte le fait que chaque objet (i) se compose de plusieurs primitives géométriques simples qui sont à tester avec celles de l'autre objet virtuel (j) ou avec celles de (i) si nous nous intéressons aux auto-collisions. Pour réduire cette complexité, les algorithmes de détection de collisions doivent procéder par étapes successives, après avoir éliminé les cas inutiles. Ainsi, les algorithmes de détection de collisions peuvent être réalisés globalement en trois phases [5]:

- **Phase accélératrice de recherche de proximité (broad phase)**
  C'est une étape de détection grossière permettant de trouver quels sont les couples d'objets susceptibles d'entrer en collision, parmi tous les couples possibles.
- **Phase accélératrice de détection approximative (narrow-phase)**
  C'est une étape de détection permettant de déterminer les zones de collision potentielle.
- **Phase précise (noyau)**
  Dans cette étape, seules les zones de détection potentielles (résultats de la phase accélératrice) sont investiguées. C'est dans cette étape que l'on opère aux détections entre paires de primitives bas niveaux avec éventuellement, l'identification de la localisation et la connaissance de l'état exact du contact et de ses paramètres (quantification).

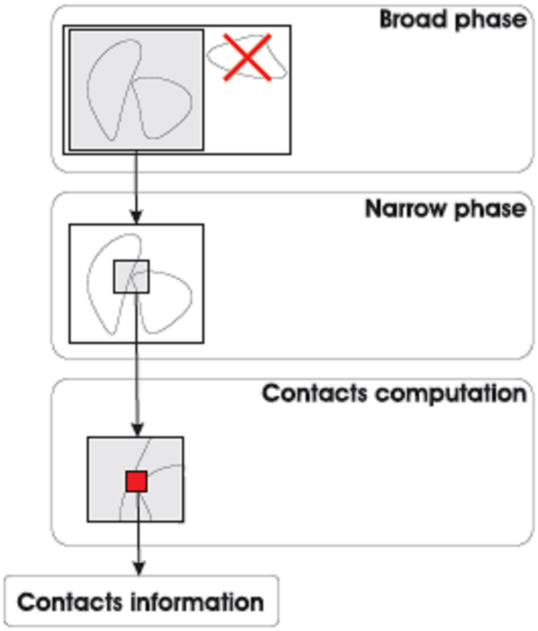


Figure I.1. Schéma standard d'un cycle de détection de collision [8]

Les deux premières tâches de ce cycle sont assimilables à des filtres plus ou moins grossiers destinés à réduire l'espace géométrique considéré lors de la phase de détermination des informations de contact. Elles vont ainsi permettre une diminution importante de la combinatoire à considérer en identifiant, dans un premier temps, les paires de modèles susceptibles d'entrer en collision (*broad phase*), puis, dans un deuxième temps, les zones géométriques (*narrow phase*) à prendre en compte pour la détermination des informations de contacts.
Une autre manière de réduire la complexité est de profiter des propriétés de la scène virtuelles dont lesquelles les objets évoluent et qui sont : la cohérence spatiale et la cohérence temporelle.

- **La cohérence spatiale**

Deux objets, très éloignés l'un de l'autre ne peuvent pas être en collision à cause de quelques règles simples de la physique. Pour profiter de cette observation, il faut subdiviser l'espace en plusieurs parties et y distribuer les objets selon leurs positions. Il n'est nécessaire alors que de chercher les collisions entre les paires d'objets se situant dans la même partie de l'espace. Dans ce cas, on réduit la complexité de l'algorithme précédent [2].

- **La cohérence temporelle**

Lorsque les acquisitions des positions sont très fréquentes par rapport à la vitesse des objets, le changement de position relative entre les paires d'objets, d'une itération à la suivante, est relativement faible. On peut alors sauvegarder une partie des résultats de l'itération actuelle, susceptible d'être utile dans la suivante. Ce genre d'information est appelé « témoin » et sert généralement comme point de départ de l'algorithme dans l'itération suivante, favorisant ainsi la recherche de solutions locales, généralement plus rapide et évitant alors de recommencer du début à chaque itération.

De ces deux propriétés, on peut tirer deux types de détection de collision [2] qui sont la détection spatiale et la détection spatio-temporelle dont les détails seront abordés ci-dessous.

2.3 Les types de détection de collision

On distingue deux grandes classes de techniques :

1- La première consiste à échantillonner les trajectoires au cours du temps et à considérer les intersections à des instants discrets. De la sorte, le mouvement est considéré comme une suite discrète de dates, en cherchant ensuite les collisions (c.-à-d. Intersections) à chaque date. Ces méthodes sont classiquement appelées méthodes de collisions à temps discret, c'est la classe des méthodes spatiales.

Soit au contraire, l'aspect continu du temps est conservé, et les intersections de trajectoires sont calculées. Ces méthodes sont dites à temps continu, ou les méthodes spatio-temporelles [9].

2.3.1 La méthode spatiale

Considérons une scène comportant N objets en mouvement les uns par rapport aux autres. La détection de collisions spatiales consiste en un test de collision entre ces N objets à différents instants t, sans tenir compte de l'aspect dynamique de la scène. La méthode la plus simple pour réaliser la détection de collision sur l'ensemble de la scène est de faire le test de collisions pour chaque paire d'objets de la scène, cette méthode a alors une complexité en $O(N2)$. Si le nombre d'objets devient très grand, alors les tests de collisions sur la scène deviennent très lents, ce qui rend le tout de moins en moins interactif [10].

L'inconvénient de cette méthode réside dans le fait que certaines collisions peuvent ne pas être détectées. S'ils sont suffisamment rapides, deux objets peuvent passer l'un à travers l'autre entre les instants où a lieu le test. L'exemple de la (figure I.2) montre le cas d'une intersection non détectée lorsque deux objets ont une vitesse trop élevée. Ni à l'instant t, ni à l'instant $t+\Delta t$ on ne détecte d'intersection et de ce fait, on ne détecte aucune collision, alors que les objets sont effectivement passés l'un à travers de l'autre [6].

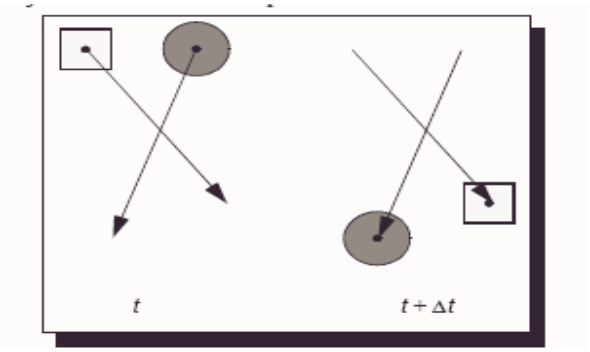


Figure I.2. Exemple de collision non détectée [6]

Finalement, le problème de la détection revient alors à un calcul d'intersection 3D qui fait appel à des méthodes connues en géométrie algorithmique, dont les résultats se classeront sous différents niveaux, en fonction de la quantité croissante d'informations retournées [6]:

- **La simple détection d'intersection vide**
  Les algorithmes obtenus ne fournissent qu'une réponse booléenne et sont alors les plus rapides si des critères certifiant la collision ou la non-collision sont trouvés ;
- **Les intersections des surfaces des objets**
  On cherche seulement les éléments d'un objet qui présentent des intersections avec la surface de l'autre objet en omettant les éléments complètement compris dans l'autre objet ;
- **L'amplitude de l'intersection des volumes**
  Par exemple le volume de l'intersection ou la profondeur de pénétration. Cette information est très utile pour le calcul de certaines réponses à la collision. Dans le cas d'un volume d'intersection non connexe, les approches peuvent être locales (on évalue l'amplitude de toutes les parties en collision) ou globales (on cherche la plus petite translation permettant de séparer les deux objets) ;
- **La forme de l'intersection des volumes**
  Elle est cependant difficile, voire parfois impossible à calculer formellement.

2.3.2 La méthode spatio-temporelle

Dans la détection spatio-temporelle, on s'intéresse au mouvement des objets dans la scène, donc on prend en compte le paramètre temps, et on cherche à déterminer à quel instant la collision a lieu. Cela peut se faire de quatre façons :

2.3.2.1 Détection des contacts à partir d'intersection tridimensionnelle 3D (détection discrète)

Cette approche tire parti de la relative simplicité des méthodes à intersection 3D, et en particulier des algorithmes de détection des intersections vides. Une fois l'intersection détectée, on revient en arrière dans la simulation. On utilise pour cela une dichotomie afin

de remonter au temps où a eu lieu l'interpénétration. Sur la (figure I.3), on suppose deux objets entrant en collision.
Au moment *t*, il n'y a pas collision. Au moment $t+\Delta t$, on constate une intersection. On cherche alors par dichotomie l'instant $t+\delta t$ où a eu lieu la collision [6].

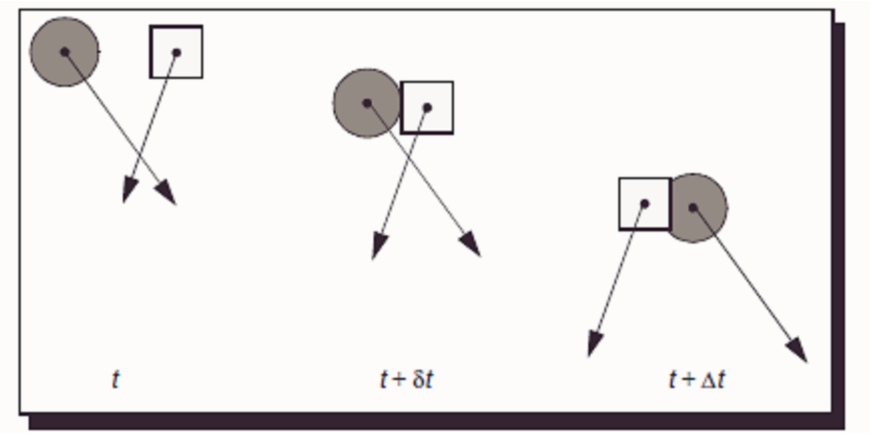


Figure I.3. Détection de collision par interpénétration **[6]**

Seulement, il est possible de ne pas détecter certaines collisions si les instants initiaux sont tels que deux objets se rencontrent entre deux instants échantillonnés sans être en collision à ces instants [10]. Afin d'éliminer les cas non détectés des approches 3D, il faut considérer l'ensemble des positions prises par l'objet au cours du temps : le temps doit alors nécessairement être considéré sous forme continue [6].

2.3.2.2 Intersection des volumes de trajectoire ou volumes balayés

On considère, dans cette solution, tout le volume décrit par les objets entre *t* et $t+\Delta t$. ensuite, on calcule l'intersection 3D de ces deux volumes. Cette méthode n'est pas satisfaisante, car ce n'est pas parce que les deux volumes se croisent qu'il y a forcément une collision. En effet, les positions des objets correspondant à l'intersection peuvent être atteintes à des instants complètement différents. La figure 4 illustre ce problème en 2D. Les deux objets ne s'interpénètrent pas durant le pas de temps et pourtant, on trouve une intersection de leurs volumes de trajectoire [6]. D'autre part, la génération du volume balayé par un objet entre deux instants donnés peut être très coûteuse en temps de calcul. De plus, le calcul de l'intersection entre les deux volumes balayés par deux objets peut aussi être très coûteux en temps de calcul à cause de la complexité des volumes balayés [10].

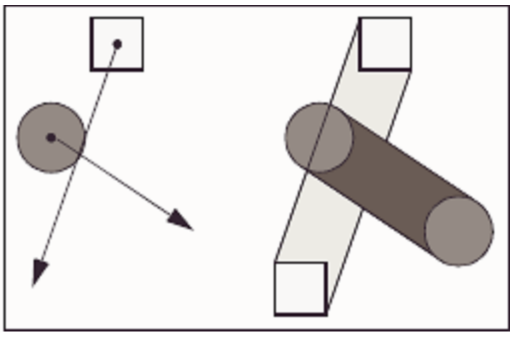


Figure I.4. Intersection de volumes de trajectoire [6]

2.3.2.3 Intersection de volumes extrudés selon le temps (Méthode par extrusion des trajectoires en 4D)

Afin de pallier le problème précédent, on peut se placer en dimension 4, c'est-à-dire considérer le temps comme une variable d'espace supplémentaire. Dans ce cas, on cherche non plus une intersection de volumes 3D, mais 4D. Le problème précédent (intersection de volumes 3D) devient alors l'intersection des projections des volumes 4D sur l'hyperplan 3D de l'espace du monde. À cause de la dimension supplémentaire, cette approche est plus délicate à mettre en œuvre que la précédente [6]. Dans le cas de formes d'objets simples, la collision peut être calculée de façon analytique à l'aide d'une équation où la seule inconnue est le temps, ainsi les solutions sont les instants où se produit la collision. Mais en général, cette approche est relativement coûteuse pour générer les volumes extrudés en 4D (par exemple, l'extrusion d'un objet linéaire qui est soumis à une vitesse de rotation constante est une hyper surface hélicoïdale). Pour cette raison, cette méthode est utilisée seulement pour des objets linéaires ne subissant uniquement que des translations seules [10].

2.3.2.4 Méthode par paramétrisation des trajectoires (méthode analytique)

Elle est strictement équivalente à la méthode par intersection de volumes 4D mais on regarde le problème sous un angle analytique plutôt que géométrique. Le solide est considéré comme incorporant un paramètre supplémentaire, le temps. Trouver s'il y a interpénétration revient à trouver une solution à un système d'équations. On essaye, dans ce cas, de réduire le système à une seule équation dépendant uniquement du temps [6]. Considérons, par exemple, le cas simple d'un point en mouvement linéaire dont nous voulons savoir s'il passe à travers d'un triangle fixe dans l'espace. Alors, l'équation paramétrique vectorielle est la suivante :

$$P + (p' - p)\, t = p_0 + (p_1 - p_0)\, u + (p_2 - p_0)\, v$$

Où p et p' sont les positions initiales et finales du point, et (p_0, p_1, p_2) définit les sommets du triangle, u et v les variables paramétriques pour le plan défini par le triangle, et t est la

variable de temps qui est à 0 lorsque le point est en *p*, et à 1 lorsque le point est en *p'*. Après résolution de l'équation paramétrique vectorielle, si :

$$0 \leq t \leq 1 \text{ et } u \geq 0 \text{ et } v \geq 0 \text{ et } u + v \leq 1$$

Alors, le point passe à travers le triangle entre les deux instants, car la droite (*pp'*) coupe le triangle. Dépendant des trajectoires, les degrés des polynômes résultants peuvent être arbitrairement élevés. Alors, comme les polynômes d'ordre 5 et plus ne peuvent pas être résolus analytiquement, la détermination de l'instant de collision peut être très coûteuse en temps de calcul pour des trajectoires quelconques [10].

2.4 Accélération de la détection de collision

En matière de détection de collisions, il existe plusieurs méthodes accélératrices qui visent à réduire la complexité du processus de détection de collisions, comme nous l'avons déjà cité plus haut, le nombre de paires d'éléments à tester dans une scène à chaque pas d'animation est très grand, surtout quand la simulation est en temps réel, ces méthodes utilisent pour la plupart d'entre elles les volumes englobants et les hiérarchies qui leurs permettent dans le cas moyen de ne tester que $O(n \times log(n))$ collisions [4].

2.4.1 Volumes englobants

a. Volumes englobants classiques

L'idée des volumes englobants est de placer chaque objet dans une forme géométrique simple, les éléments dont les boites ou volumes englobants ne sont pas en collision, ne peuvent être en collision, ceci pour éliminer les calculs inutiles. Plusieurs types de géométries peuvent être utilisées, les plus communément utilisées sont les sphères, les boîtes isothétiques de même orientation, c'est-à-dire alignée sur des axes (AABB : Axis-AlignedBounding Boxes), les boîtes englobantes orientées ou à orientation quelconque (OBB : OrientedBounding Boxes). Cependant, d'autres volumes plus complexes sont exploitables comme les polytopes à orientation discrète (*k*_DOP), les *tribox* ou encore les enveloppes convexes des objets [5].

La méthode d'accélération basée sur les volumes englobants appartient à la phase accélératrice de détection approximative (narrow phase) qui permet d'identifier les zones géométriques à prendre en compte pour la détermination des informations de contacts [5].
Chacun de ces types de volumes englobants se voit plus ou moins adapté en fonction de la géométrie considérée et du niveau de performance escompté. La détection de non-intersection se base sur le principe de trouver un axe séparant les deux objets. Le choix de l'axe est conditionné par le volume choisi. Les intersections de sphères et d'AABB sont les plus rapides. Ainsi, les sphères englobantes utilisent l'axe entre les centres des objets, elles ont pour principal avantage la simplicité des tests de chevauchement. En effet, un test de chevauchement entre deux sphères se limite à une comparaison entre la distance séparant leurs centres et la somme de leurs rayons respectifs. Cependant, les sphères donnent

généralement une approximation moins optimale de la géométrie considérée, suivies de près par les AABBs qui choisissent les axes du repère global, et les *k*-DOPs qui, quant à eux y ajoutent des axes supplémentaires. Contrairement, le test entre OBB est plus lent, mais ces volumes offrent une meilleure approximation des objets, de même pour les enveloppes convexes. Ce compromis fait que le choix du volume dépend beaucoup du contexte et en particulier de la forme et du mouvement des objets considérés. La figure I.5 représente une illustration de ce phénomène [5][8][6].

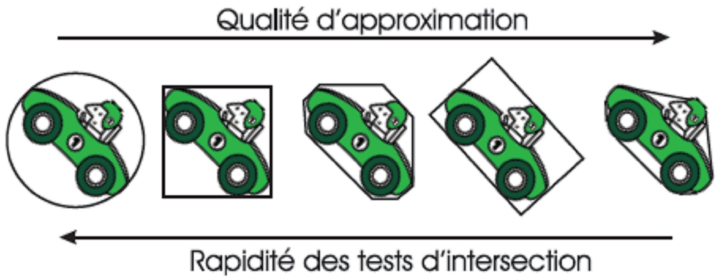


Figure I.5. Comparaison entre différents types de volumes englobants de gauche à droite : sphère, AABB, 8-Dop, OBB, enveloppe convexe [8]

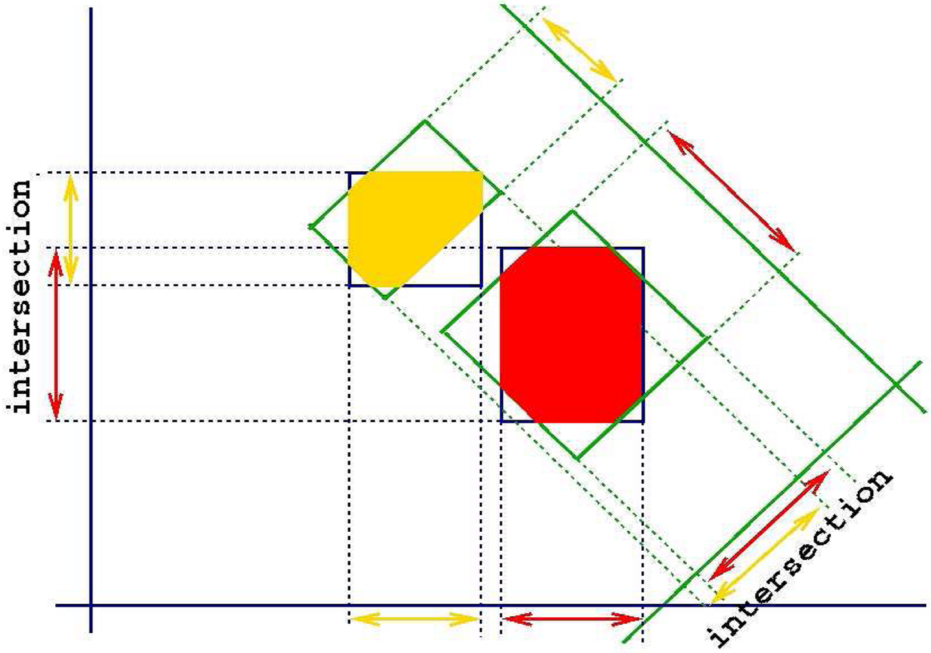


Figure I.6. Le test d'intersection entre K-Dop [4]

Il suffit de tester si les projections orthogonales des K-Dops dans tous les repères génèrent des intersections.

b. K-Dop orienté et gonflé.

C'est un K-Dop gonflé par rapport au K-Dop minimal couvrant l'objet, ensuite il est orienté vers la direction du mouvement de l'objet qu'il englobe. Le but de ces deux manipulations étant de laisser une « marge de détection » et ne pas oublier une collision qui s'est déroulée entre deux pas de temps. En effet, la discrétisation du temps pose un problème qui réside dans le fait qu'entre deux pas de temps, on ne peut rien analyser ; il faut donc des pas de temps les plus petits possible. Cependant, plus ces pas sont petits, plus il faut faire de calculs. Il faut donc trouver le bon compromis entre bonne discrétisation du temps et temps réel. La (figure I.7) représente une inflation orientée d'un 8-Dops.

Grâce à sa marge de détection, cette méthode génère de bons résultats dans la détection des collisions, avec une certaine efficacité, car le calcul d'intersection entre K-Dops est très rapide, mais son temps de traitement reste encore trop long pour être utilisable en temps réel [4].

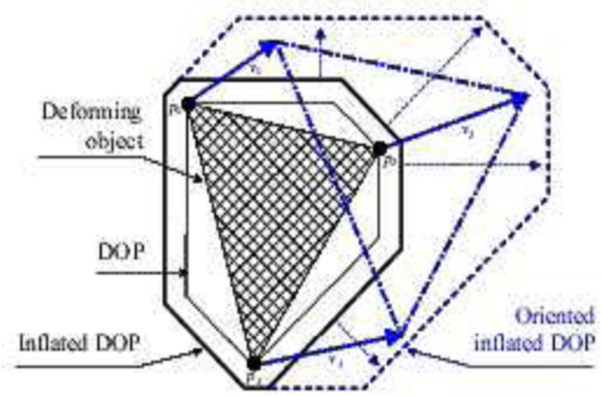


Figure I.7. Estimation du mouvement et inflation orientée d'un 8-Dop[4]

2.4.2 Les hiérarchies

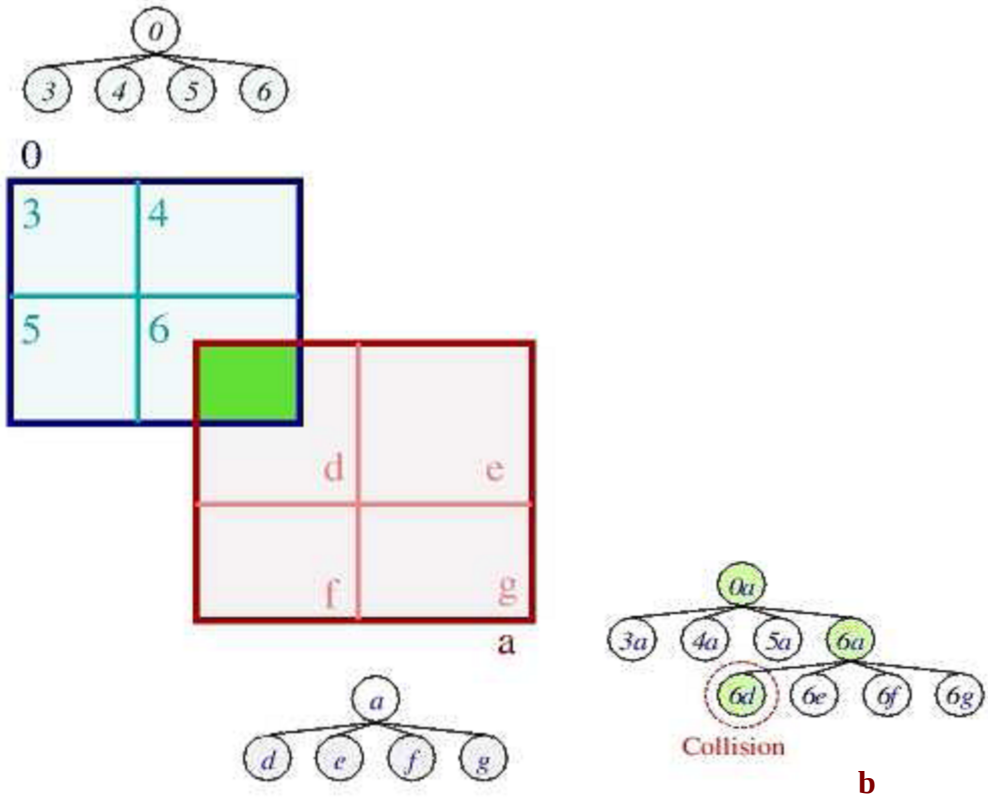
Une hiérarchie permet de regrouper dans un même ensemble les éléments géographiquement proches. Son but étant de faire des tests de collisions sur les volumes englobants pour éliminer, par blocs, des ensembles de paires d'éléments qui ne peuvent pas être en collision. On commence par tester entre les plus gros volumes et s'il y a collision, on descend d'un niveau et on itère. Un élément important dans ce type d'algorithmes est le choix de la construction de l'arbre représentant la hiérarchie, et l'algorithme de parcours de celui-ci [4].

Une hiérarchie de volumes englobants peut se construire selon deux approches [8]:

- **Top-down**
  Cette approche consiste en une construction de la hiérarchie du haut vers le bas. Dans un premier temps, un volume racine englobant l'ensemble du modèle est calculé. Celui-ci est par la suite décomposé en n sous-volumes ($n \geq 2$). La décomposition se poursuit récursivement sur les sous-volumes obtenus jusqu'à obtenir les volumes *feuilles* contenant chacun une primitive élémentaire du modèle (par exemple un triangle dans le cas polyédrique).

- **Bottom-up**

  Cette approche consiste en une construction de la hiérarchie du bas vers le haut. Dans un premier temps, l'ensemble des volumes feuilles contenant les primitives élémentaires du modèle vont être calculés. Ces volumes vont par la suite être regroupés selon des critères de proximité afin d'obtenir des volumes englobants de niveau supérieur. Le *regroupement* se poursuit récursivement sur l'ensemble des volumes nouvellement calculés jusqu'à obtenir le volume englobant racine contenant l'ensemble du modèle.



***a**) Deux hiérarchies représentées par des arbres quaternaires.*

***b)** Les tests effectués entre les volumes englobants pour détecter la collision.*

Figure I.8. Exemple de représentation hiérarchique avec un arbre quaternaire et de parcours en cas de collision [4]

2.4.3 Partitionnement des objets

On peut diviser les objets en plusieurs volumes englobants en créant une hiérarchie où la boite la plus grande englobe entièrement l'objet et on descend jusqu'à l'encapsulation des éléments formant la scène. On commence donc par tester sur la globalité de l'objet et on raffine de plus en plus seulement là où il y a des collisions [4], la (figure I.9) est un exemple de partitionnement d'objets avec une hiérarchie de sphère.

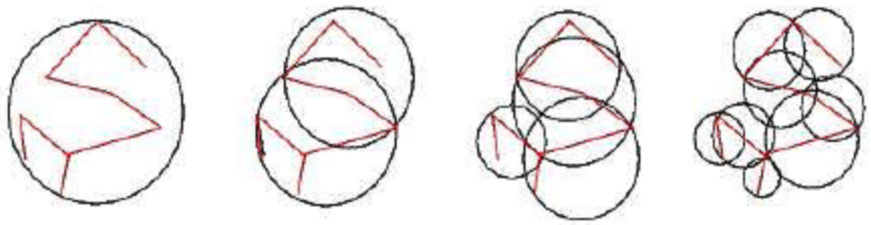


Figure I.9. Partitionnement d'objets, exemple d'une hiérarchie de sphères [4]

2.4.4 Partitionnement de l'espace

Cette technique consiste à subdiviser l'espace géométrique en un ensemble de régions afin d'être en mesure d'identifier les paires d'objets susceptibles d'entrer en collision. En pratique, les algorithmes basés sur ces approches vont dresser, pour chaque région, la liste des éléments géométriques présentant une intersection non nulle avec le volume défini par celle-ci. Ainsi, seules les paires constituées d'éléments présents dans une même liste (c.-à-d. dans une même zone de l'espace) seront prises en considération lors de la détection de collision. La subdivision de l'espace peut être réalisée de différentes manières [8]. Dans la suite, nous présentons ces différents types de représentations.

2.4.4.1 Les Grilles

En partitionnant l'espace, les volumes englobants auront une position et une dimension fixe dans l'espace ; on obtient la hiérarchie en divisant l'espace par une grille de plus en plus fine. Si la grille est uniforme, la division produit des régions de taille unique. Chaque objet est alors associé à une grille s'il est complètement contenu dans une seule, ou à plusieurs grilles s'il se trouve sur la frontière entre ces grilles. Une des difficultés de l'uniformité de la grille est le choix de la taille. Si elle est très petite, la structure de données aura besoin d'un très grand espace mémoire. En plus, un objet de grande taille va déborder sur plusieurs grilles. Dans la (figure I.10), seules les paires d'objets B-C, et C-D sont retenues pour une détection de collision précise. Un meilleur choix est d'utiliser des hiérarchies de grilles. Un grand objet sera donc placé dans le plus petit nombre de régions pouvant le contenir (figure I.11). Il faut préciser que lorsque les objets bougent dans la scène, leurs appartenances aux grilles doivent être mises à jour, avant de chercher les collisions entre les objets de la même grille [2].

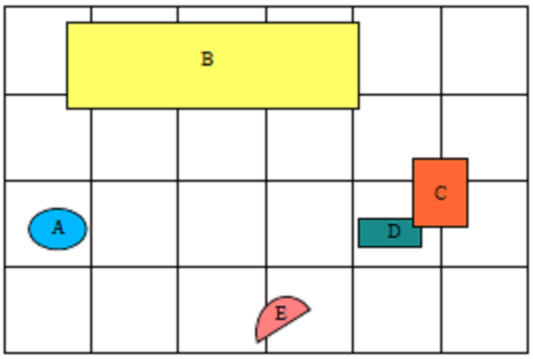


Figure I.10. L'espace divisé par une grille uniforme [2]

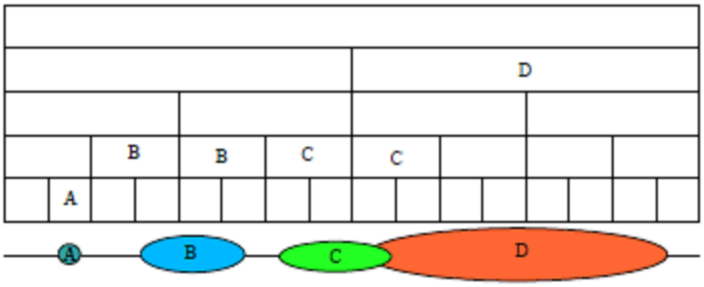


Figure I.11. Une grille en hiérarchie pour représenter un espace 1D [2]

2.4.4.2 Les Octrees

Les arbres Octrees divisent l'espace d'une manière uniforme. L'espace du départ est associé au premier nœud de l'arbre. Cet espace est ensuite divisé en 8 parties et chacune est attribuée à un enfant du premier nœud. La même procédure est employée sur chacune des parties obtenues. Là aussi, les objets sont attribués à des nœuds de l'arbre, et cette attribution doit être mise à jour lors du changement des positions des objets. Un test de collision plus précis est alors effectué sur chaque paire d'objets attribués au même nœud [2].

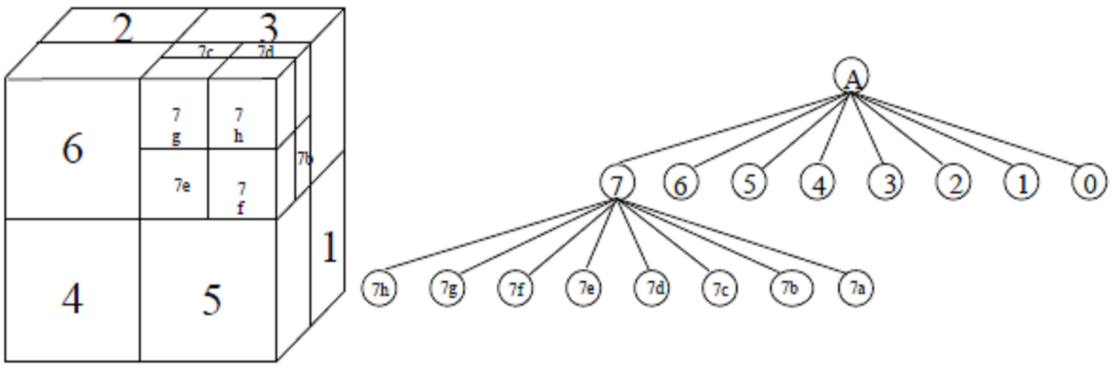


Figure I.12. Un espace divisé et son arbre Octree correspondant [2]

2.4.4.3 Les Arbres BSP

La méthode utilisée à chaque étape des BSP est la division de l'espace 3D en deux parties par un plan. Le premier sous-espace est alors dit positif, et le deuxième négatif selon sa position par rapport à la normale du plan de division. À leurs tours, chaque sous-espace est divisé par un autre plan, donnant naissance à deux nouvelles parties. Ainsi, les espaces ainsi produits représentent une hiérarchie et à chaque division deux nouvelles régions sont créées. Il existe plusieurs méthodes pour choisir le plan diviseur. L'une de ces méthodes divise l'espace en utilisant les faces des objets alors que d'autres méthodes choisissent le plan selon d'autres critères de la scène. Il faut noter que la création d'un arbre BSP prend beaucoup de temps et se fait généralement hors ligne. Lorsque les objets bougent, il n'est pas possible de modifier l'arbre correspondant durant la simulation, il est alors logique d'utiliser les BSP pour les objets qui ne vont pas changer de position [2]. La figure I.13 présente un espace divisé et son arbre BSP correspondant.

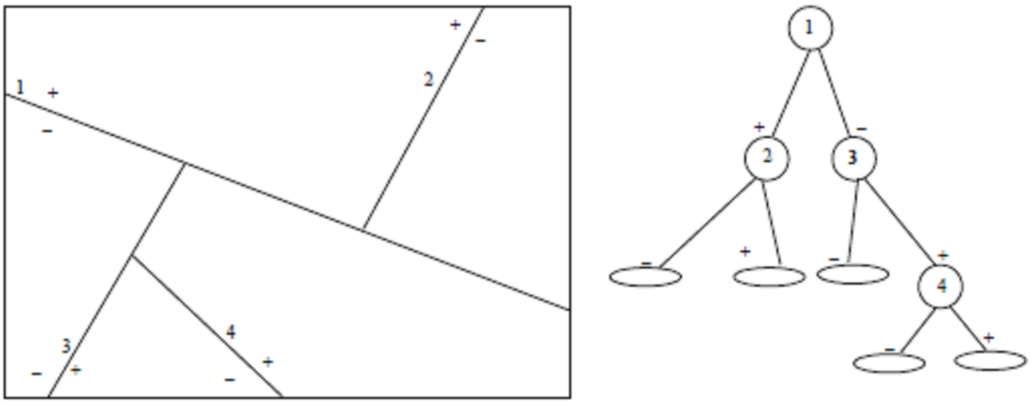


Figure I.13. Un espace divisé et son arbre BSP correspondant [2]

2.4.4.4 Le Sweep and prune

Le « sweep and prune » est une méthode introduite par Baraff [11], et a été utilisée dans plusieurs logiciels de détection de collisions. L'algorithme se base sur l'idée suivante: si deux objets sont en collision, alors leurs projections sur les trois axes ont obligatoirement des parties communes. Autrement dit, si les projections sur un des trois axes n'ont pas de parties communes, alors les objets ne sont pas en collision. Pour mettre cette idée en œuvre, il faut procéder en trois étapes :

- On détermine pour chaque objet de la scène, les valeurs minimales et maximales sur chaque axe x, y, et z. (cela revient à déterminer une AABB autour de l'objet).
- Les limites des intervalles de chaque axe sont placées dans une liste séparée, et ces valeurs sont ordonnées par ordre croissant. À ce moment, nous avons trois listes correspondant aux trois axes, chacune contenant les valeurs minimales et maximales de chaque objet de la scène sur l'axe correspondant.

- Pour trouver les paires d'objets susceptibles d'être en collision, il suffit de parcourir la première liste (liste des x par exemple) pour chercher les objets qui ont une partie d'intervalle commune. Lorsqu'une telle partie est trouvée sur le premier axe, les intervalles sur le deuxième et le troisième axe doivent être vérifiés. Si les trois parties présentent des intersections, alors les deux objets correspondants doivent être testés plus précisément. Un exemple en 2D est présenté dans la figure I.14. Dans ce cas, seules les paires A − B et D − E seront retenues, puisque leurs projections sur les deux axes ont des parties superposées [2].

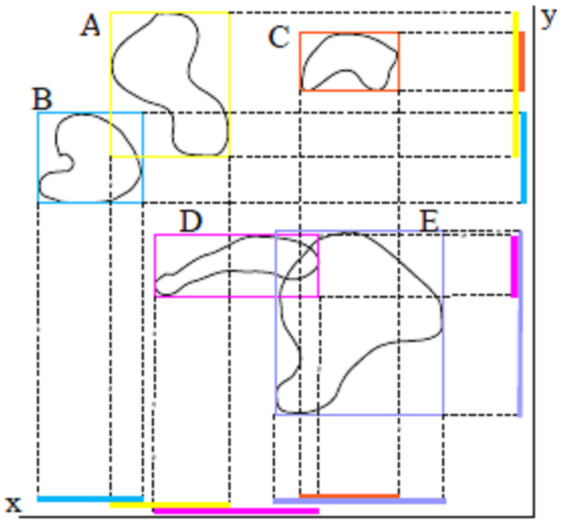


Figure I.14. Sweep and prune en 2D [2]

2.4.5 La multi résolution

Il s'agit de mettre en œuvre une hiérarchisation duale des composantes géométriques des objets. Chaque objet est décomposé en une hiérarchie de volumes englobants dont chaque nœud se voit correspondre à plusieurs représentations de résolution variable de la zone géométrique à laquelle il est associé (on parle de *CLODS : Collision Levels of Details*). Aussi, cette technique permet d'effectuer la détection de collisions de façon plus ou moins précise en fonction de certains critères (planéité… etc.) visant à garantir un comportement des objets réalistes, particulièrement dans le cadre de simulations mettant en œuvre un système de manipulation haptique [8].

Figure I.15. Exemple de représentation multi résolution [8].

2.4.6 L'Accélération matérielle

Ce sont des approches visant à exploiter des calculateurs hardware, et particulièrement les unités de calculs dédiées à la restitution graphique (on parle de *GPU*).

Les approches basées sur ce type de matériel imposent l'utilisation de géométries de nature polyédrique. Elles sont classifiables en deux catégories [8] :

- **Approches basées sur les triangles**

  Ces approches se basent sur la récupération des triangles appartenant au cône de visée (*view volume*). Elles sont limitées par la géométrie du *view volume* qui, au sein des cartes graphiques actuelles, ne peut être que de deux types : boîte parallélépipédique ou pyramide tronquée à base rectangulaire.

- **Approches basées sur les pixels**

  Ces approches, dites basées sur l'image, exploitent le *pixel shader*. En d'autres termes, elles ne se basent pas sur les triangles proprement dits, mais sur leur représentation par pixels. Il est à noter que la précision de la détection est ici liée à la taille des pixels. Aussi, un compromis entre niveau de résolution et temps de calcul est à considérer. Des techniques plus récentes également basées sur ce type d'approches utilisent les nouvelles fonctionnalités proposées dans les cartes graphiques les plus évoluées telles que la détection d'occlusions. Les performances se voient ainsi grandement améliorées.

2.5 Algorithmes calculant la distance entre deux objets

La détection de collisions se base principalement sur le calcul de la distance entre deux objets. Quand cette distance est nulle, les deux objets sont en contact. La distance est dite *négative* si les deux objets s'interpénètrent virtuellement. La valeur de cette distance négative peut être obtenue en calculant le déplacement minimal qui permet de séparer les deux objets [1].

2.5.1 L'algorithme GJK

L'algorithme Gilbert-Johnson-Keerthise se base sur une méthode itérative de calcul de distance entre objets convexes, son avantage réside dans sa simplicité qui le rend facile à implémenter. Contrairement aux autres algorithmes, l'algorithme GJK n'exige pas que les données géométriques soient stockées dans un format spécifique, mais repose plutôt sur une fonction de support pour générer itérativement des simplexes proches de la réponse correcte à l'aide de la somme de Minkowski (CSO) de deux formes convexes. [23]

D'un point de vue mathématique, la distance entre deux points peut être mesurée comme la différence entre les deux vecteurs, générés à partir de l'origine et les deux points. Si ces derniers étaient les ensembles de points A et B du calcul de la différence entre les points de A et de B, nous serons en mesure de trouver la distance minimale entre les deux ensembles de points.

D'un point de vue informatique, le calcule de toutes ces différences consomme du temps de calcule et des ressources, l'algorithme GJK peut dépasser cet inconvénient [23].

L'idée de base dans cet algorithme, sans entrer dans les détails de ce dernier, est de calculer la distance entre les deux enveloppes convexes de deux ensembles de points A et B et donner la valeur de cette distance dans le cas où les enveloppes convexes sont séparées, et une approximation de la distance négative dans le cas contraire [1].

Cet algorithme a connu diverses améliorations. Les travaux de Cameron [17] accélèrent la recherche des sommets intervenant dans les simplexes par *hill-climbing* (on cherche à minimiser des produits scalaires par cohérence spatiale) et proposent d'exploiter la cohérence temporelle pour établir des conditions de départ faisant converger le calcul plus rapidement. Dans [18], Baraff propose d'étendre l'algorithme de deux façons. Il devient plus robuste pour les objets convexes non polyédriques en calculant une borne inférieure de la distance et en contrôlant l'erreur. Par ailleurs, il y'a à noter l'augmentation de sa rapidité en fournissant une méthode directe pour calculer les points des simplexes les plus proches de l'origine [6].

2.5.2 L'algorithme de Joukhadar et Laugier

Cet algorithme s'appliquant aux objets déformables non convexes calcule la distance négative entre les deux enveloppes convexes de deux ensembles de points X et Y. Il donne une mesure de la pénétration dans le cas où les enveloppes convexes sont en collision.
Selon cet algorithme, pour trouver la distance négative, on procède de la manière suivante :

1. On commence par la séparation de X et Y selon la dernière direction du contact *N*.

2. On applique l'algorithme de GJK afin d'obtenir une nouvelle valeur de la direction du contact N'.

3. Si $N==N'$ la distance négative est $|N| - |N'|$ Sinon, $N== N'$...et on recommence à l'étape numéro 1.

Cet algorithme fait plusieurs itérations pour déterminer la direction du contact. A chaque itération *i*, la valeur de N_i est plus proche de $|N_{i-1}|$, *il est* donc convergent. La complexité de cet algorithme est *k* fois la complexité de l'algorithme de GJK, où *k* est le nombre d'itérations nécessaires pour converger. Donc, la complexité de l'algorithme est $0\{k\,(V_a + V_b)\}$ [1].

2.5.3 L'algorithme de Lin et Canny

On considère A et B deux polyèdres convexes, l'algorithme de Lin et Canny consiste à déterminer les éléments (*i.e.* sommet, arête, facette) de A et de B permettant d'obtenir la plus petite distance. À partir d'un couple (f_A^0, f_B^0) d'éléments de A et B, on cherche par utilisation des régions de Voronoï entourant A et B un nouveau couple (f_A^1, f_B^1) plus proche. Cet algorithme est répété itérativement jusqu'à ce qu'un minimum local soit trouvé (qui est un minimum global grâce à la convexité de l'objet). Cet algorithme peut être optimisé en mémorisant le dernier couple trouvé. La cohérence temporelle 1 permet

alors d'espérer une convergence en temps constant O (1). Une fois que ce couple est trouvé, la distance euclidienne entre les deux objets est la plus petite distance entre le couple d'éléments trouvés [14].

2.5.4 Algorithme de Garcia-Alonso et al

L'idée de cet algorithme [11] est de représenter l'objet par plusieurs niveaux de précision. Le premier niveau est un parallélépipède dont les facettes sont parallèles aux axes des coordonnées. On appelle ce parallélépipède 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 I.16) qui est le parallélépipède 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 (volumes élémentaires). Un voxel est mis à 1 s'il contient une partie de l'objet et à 0 sinon.

Quand les deux polyèdres sont assez éloignés 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 ou 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 couple de voxels, on détecte le contact entre eux et on détermine les facettes qui leur correspondent.

Le container d'un objet, la matrice de voxels, la relation entre les voxels et les facettes de l'objet peuvent être déterminés 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].

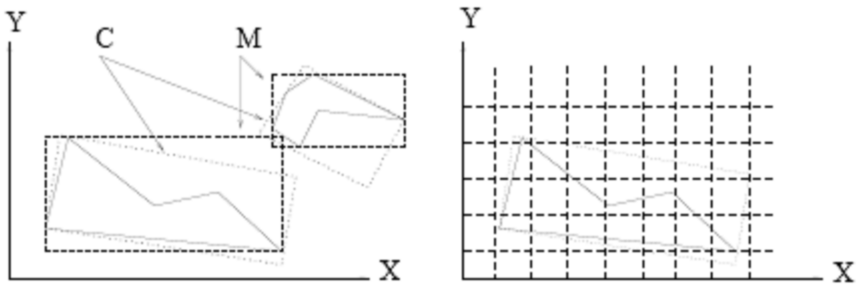
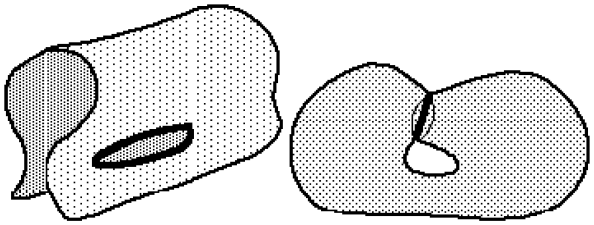


Figure I.16. Les boites min-max, les containers et les voxels [1]

2.5.5 Algorithme de Volino & Thalmann

Cet algorithme détecte la collision d'un objet surfacique déformable avec lui-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. I.17.(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 I.17. (b)). Une surface ne peut pas avoir deux facettes en collision si sa projection selon V ne représente pas une auto-collision.



(a) (b)

Figure I.17. Auto-collision due à : (a) la courbure (b) la forme du contour [1]

Pour trouver le vecteur $\vec{V}$, Volino et Thalmann proposent un algorithme récursif [15]. Ils commencèrent par une famille initiale de vecteurs qui représente une discrétisation de l'espace. Pour chaque facette, ils cherchèrent les vecteurs qui donnent un produit scalaire positif avec cette facette et propagèrent ces informations récursivement (voir figure I.18). Le temps nécessaire pour trouver $\vec{V}$ est donc de l'ordre de $O(n\log_2 n)$ où *n* est le cardinal de la famille initiale de vecteurs [1]. On peut trouver récursivement le vecteur $\vec{V}$, dont les produits scalaires avec toutes les normales aux facettes sont positifs.

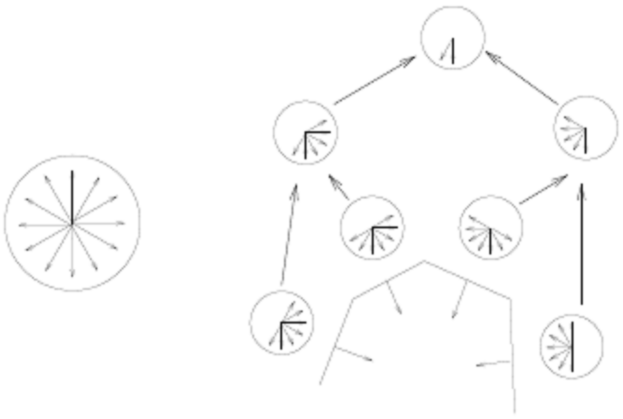


Figure. I.18. Recherche récursive du vecteur $\vec{V}$ [1]

2.5.6 Algorithme de Baraff & Witkin

Pour traiter des objets déformables, Baraff et Witkin [19] divisent statiquement chaque objet en plusieurs sous-objets dont les déformations pendant le mouvement sont polynomiales de premier degré. Cela garantit que les sous-objets ne passent 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].

2.6 Traitement de collision

Dans la partie précédente de ce chapitre, nous avons présenté la détection des collisions. La détection des contacts entre les objets est la partie la plus cruciale pour la plupart des algorithmes géométriques, et en particulier pour la simulation dynamique, mais après la détection de ces collisions, il faudra pouvoir déterminer comment ces objets en collision vont réagir [1]. Nous présentons ci-après le traitement de collision ainsi que les différentes méthodes utilisées.

2.6.1 Définition

La réponse aux collisions vise à reproduire les interactions entre les objets d'une façon précise et réaliste. Elle consiste à trouver les nouvelles positions et vitesses des objets en collision et ceci dans le but d'éviter les interpénétrations irréalistes des objets et de simuler des effets réalistes de rebondissement et de frottement [13].

2.6.2 Les méthodes de traitement de collisions

Dans le domaine de la simulation physique, il existe principalement trois méthodes pour traiter les collisions, à savoir : la méthode des contraintes, la méthode des pénalités et la méthode des impulsions.

2.6.2.1 Méthode des contraintes

Ce sont les méthodes les plus intuitives [3], elles consistent 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 : [1]

$$p(t)=f(force) \qquad (1)$$

$$C(p(t))>=0 \qquad (2)$$

L'équation *(1)* caractérise le mouvement de chaque objet en fonction des forces externes.
L'équation *(2)* représente les contraintes de non-pénétration entre ces deux objets.

Supposons qu'un point entre en collision avec une surface. On fait le bilan des forces appliquées au point de la collision, et on annule la composante de la force qui est normale à la surface si cette composante est orientée dans la même direction. Cela revient à considérer que de la surface part une force de réaction empêchant l'interpénétration du point à travers la surface. L'accélération de la particule est donc cantonnée au plan tangent à la surface tandis que l'on force la vitesse normale à 0. C'est la méthode couramment employée dans les systèmes physiques où l'on fait le bilan des forces [3].
Dans [20], les auteurs calculent les forces de réaction aux sommets de la zone polygonale de contact grâce à des contraintes à respecter. Il en résulte un problème d'optimisation quadratique. Ils proposent alors une heuristique consistant à deviner les points de contact quittant la surface. Le problème devient alors linéaire et est résolu par LCP (LinearConstraineProgramming).

Cette méthode n'est pas complètement satisfaisante, notamment pour prendre en compte les frottements. Dans [21], l'auteur reformule le problème et obtient un algorithme de résolution itératif. Cependant, la terminaison de l'algorithme n'est pas prouvée dans le cas des contacts avec frictions statiques [6]. Enfin, les travaux de Baraff [19, 22] cherchent à étendre, aux corps déformables, les méthodes déjà utilisées pour les corps rigides, mais cette solution n'est applicable qu'aux corps à jeu de déformations linéaires [6].

2.6.2.2 Méthode des impulsions

Les méthodes basées sur la notion de la force *d'impulsion* : l'impact 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-mêmes. 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 restitution. Si $e=0$, la collision est parfaitement élastique et si e =1, la collision est parfaitement plastique. E peut également être calculé par l'hypothèse de Poisson qui dit :

$$e = \frac{Pr}{Pc}$$

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 ne prend pas explicitement en compte 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 considère la condition que les objets entre deux collisions sont en mouvement balistique. Cette

condition permet d'optimiser le temps de calcul car le temps de la résolution de la collision n'est pas négligeable [1].

2.6.2.3 Méthode des pénalités

Cette méthode consiste à générer des forces répulsives entre les deux objets en collision pour les empêcher de se pénétrer davantage. Cette méthode nécessite de générer des forces de haute résistance pour éviter les collisions, rendant les équations différentielles de la simulation très raide ce qui peut provoquer l'instabilité dans la simulation lors du traitement par le solveur numérique [12].

Une mise en œuvre simple de cette idée est facile à réaliser : sitôt que deux objets entrent en collision ou sont très proches l'un de l'autre, un ressort est ajouté créant une force tendant à éloigner les deux corps. Cette méthode présente deux inconvénients principaux: tout d'abord, il est difficile de régler la constante de raideur du ressort car quand on l'augmente, on ajoute une autre fréquence au système d'où d'éventuels problèmes de convergence des équations. Par contre, si on la diminue, les chocs sont lents, peu énergétiques et l'interpénétration dure longtemps. Le second inconvénient est un mauvais traitement des contacts prolongés. Ces derniers ne sont pas stables et tremblent à cause des ressorts sur la zone de contact [3].

3. Auto-collisions

En général, la détection des auto-collisions peut exploiter les méthodes de détection des collisions standards. Mais plus ou moins elle nécessite de prendre en compte les liens de voisinages et la juxtaposition des parties de l'objet [5]. En effet, l'objet sera généralement considéré comme un ensemble d'éléments (polygones, triangles ou tétraèdres…), même en exploitant cette propriété qui nous permet de prendre en compte la topologie du maillage, considérant qu'une arête commune entre deux triangles voisins n'est pas une collision arête -arête. De même pour les sommets des facettes voisines, on ne peut pas gérer bien les collisions car il se peut que deux polygones juxtaposés puissent effectivement entrer en collision s'ils se replient l'un sur l'autre [3].

Dans [15], les auteurs ont proposé un algorithme de détection d'auto-collision pour les surfaces polygonales en mouvement hautement discrétisées, basé sur les propriétés de la régularité des formes géométriques permettant d'éviter des tests de collision inutiles. Ils ont utilisé une meilleure représentation hiérarchique pour la surface, qui approche une optimisation inhérente à la hiérarchisation. Ceci a permis d'avoir les informations d'adjacence pour appliquer efficacement les optimisations géométriques. Ils ont réduit le temps de calcul en construisant automatiquement la structure hiérarchique en tant que tâche de prétraitement. La robustesse de l'algorithme a été améliorée dans [16]. Ainsi, à partir de configurations où il n'y a pas intersection ou au contraire des situations inconsistantes, les auteurs cherchèrent un algorithme permettant de conserver ou rétablir une situation consistante. Le point délicat à régler concerne le dépistage des orientations incohérentes et la détection du sens des collisions.

Dans [24], les auteurs ont appliqué un algorithme appartenant à la catégorie des algorithmes dits préventive, c.-à-d. qui ne permet pas que l'auto collision survienne. Ils ont essayé de résoudre le problème des auto-collisions en évitant le test entre sommets et triangles et les tests arête-arête. Ils ne considèrent que l'ensemble des paires de particules en tenant ces dernières loin les unes des autres pour éviter les artefacts dus aux intersections des triangles non détectés. Afin de maintenir les particules éloignées, ils ont ajouté des contraintes entre celles qui sont très proches. Ces contraintes structurelles sont semblables à celles utilisées pour éviter la compression du tissu. Un test d'accélération est utilisé en construisant une hiérarchie de volumes englobants les triangles et de tester la hiérarchie avec elle-même pour la détection de collisions. Puisqu'ils ne considèrent que l'ensemble des particules, la hiérarchie se base sur ces dernières. Les auteurs ont jugé leur approche faisable une fois qu'ils utilisent une triangulation fine et la distance préservée dépendant de la taille des triangles.

Dans [25], les auteurs ont présenté un algorithme de détection efficace d'auto-collisions pour les simulations de corps déformables utilisant des unités de traitement graphiques programmables (GPU). L'approche proposée emmagasine une représentation d'un maillage triangulaire d'un modèle déformable comme une texture 1D et détecte rapidement les auto-collisions entre toutes les paires de primitives triangulaires utilisant la capacité de parallélisme SIMD programmable des GPU.

4. Conclusion

La gestion de collisions est considérée comme un facteur important garantissant le réalisme d'une simulation. C'est un procédé lent qui nécessite des stratégies très élaborées pour être efficaces dans un contexte temps réel.

À travers ce chapitre, nous avons proposé une présentation des techniques et concepts les plus communément utilisés dans le cadre de la gestion de collisions, en observant l'intense activité de recherche qu'engendre ce sujet et qui fait évoluer les solutions algorithmiques inhérentes à la détection des collisions.
Nous avons commencé par une présentation de la gestion de collisions qui comporte la détection de collisions, sa définition ces types, ainsi que les différentes méthodes accélératrices de la détection de collisions. Nous avons vu ensuite les différents algorithmes calculant la distance entre deux objets dont les méthodes de détection se basent principalement. Nous avons vu ultérieurement le traitement de collisions et ses différentes méthodes et enfin, nous avons donné un aperçu sur la notion d'auto collisions.
Le chapitre suivant sera consacré à une étude englobant les différents travaux concernant la gestion de collisions dans le domaine de l'animation de vêtements.

Chapitre II

Simulation de vêtements

1. Introduction

L'étude de tissu a commencé dans les années 1930 dans l'industrie du textile, leurs recherches étaient concernées par les propriétés physiques du tissu, tandis que la communauté des infographes été intéressés par la création de simulations visiblement crédibles avec des temps de calcul réalistes.

Les tissus sont des matériaux viscoélastiques complexes, ils doivent avoir une résistance suffisante et en même temps être souples, élastiques et faciles à plier et à manipuler. Leur simulation n'est pas facile, car leur comportement est difficile à décrire et à prédire. Les recherches dans le domaine de la simulation du tissu ont permis, non seulement de simuler de simples vêtements statiques, mais aussi des vêtements complexes en mouvement dynamique. La simulation précise des vêtements n'est pas due seulement aux modèles de calcul avancés, mais aussi aux paramètres d'entrée exactes qui jouent un rôle important dans la reproduction correcte du comportement du tissu [34].

Dans ce chapitre, nous allons présenter la simulation de vêtements, expliquant ses différents types ainsi que les méthodes utilisées, en commençant par un bref aperçu sur les propretés mécaniques du tissu.

2. Propriétés mécaniques du tissu

D'un point de vue mécanique, le comportement des matériaux textiles est influencé par quatre facteurs principaux. Ces composants permettent la création d'une grande variété de textiles, qui répond aux exigences élevées de l'industrie du vêtement. Parmi ces facteurs, nous pouvons citer la matière première qui précise l'élasticité [34]. A titre indicatif, pour la matière première naturelle, les fils qui proviennent de fibres végétales sont moins élastiques que ceux provenant de fibres animales. Nous pouvons citer également, la structure du fil et l'assemblage des fibres composant le fil qui est responsable de la non- linéarité [35]. Ensuite vient la structure plane qui est la manière dont les fils sont assemblés qu'ils soient tissés,

tricotés ou bien collés. Par exemple, les textiles tissés sont plus rigides que ceux tricotés. Et enfin, il y a le traitement de finition qui peut affecter les fibres, les fils, et même le textile en changeant toutes les propriétés mécaniques. En ce qui concerne les systèmes informatiques, les propriétés mécaniques des surfaces déformables comme le tissu sont regroupées en quatre familles :

- Élasticité : elle caractérise les forces internes résultant d'une déformation géométrique donnée.
- Viscosité: elle englobe les forces internes résultant d'une vitesse de déformation donnée.
- Plasticité: Permet de décrire la manière dont les propriétés évoluent en fonction de la déformation.
- Résilience: qui définit les limites à partir desquelles la structure se brise.

Les plus importantes sont les propriétés d'élasticité, comme la traction, le cisaillement et la flexion qui sont principalement responsables des effets mécaniques et de l'évaluation d'un vêtement. Pour une simulation dynamique précise des vêtements, les paramètres doivent être extraits à partir de mesures véritables de tissus. Ce n'est pas une tâche simple à réaliser parce que le tissu est un matériau très complexe, qui possède un comportement isotrope. Pour cela, des précautions particulières doivent être prises lors de l'exécution des mesures et leur application à une simulation virtuelle [34].

3. Méthodes de simulation

Les différentes approches infographiques pour la modélisation des tissus ont été principalement axées sur la simulation des formes complexes et des déformations du tissu en 3D [31]. Les techniques qui ont été formulées et étudiées s'alignent principalement sur deux grands axes [33] :

- Les modèles géométriques : qui cherchent à reproduire l'aspect visuel d'une représentation fixe des tissus sans chercher à quantifier des grandeurs caractéristiques du comportement physique du tissu.
- Les modèles physiques : qui partent de lois de comportements mécaniques des tissus afin de prédire sa forme et simuler son comportement dynamique.

3.1 Modèles géométriques

Les modèles géométriques ne considèrent pas les propriétés physiques du tissu. Ils sont considérés comme des techniques qui produisent des résultats rapides, mais qui ne sont pas capables de produire le comportement dynamique du tissu. En outre, les techniques géométriques ont besoin considérablement de l'intervention de l'utilisateur. Elles sont considérées comme une forme avancée d'un outil de dessin [27].

En 1986, Weil a été le premier dans la communauté graphique à avoir cherché à modéliser la géométrie du vêtement. Il s'intéressa à des pièces de tissu rectangulaires suspendues à des points fixes (points contraints). Dans un premier temps, le tissu est approximé par une surface formée de triangles. On procède ensuite à la subdivision des triangles avec une certaine méthode, et ce, pour obtenir une forme grossière du tissu. Celle-ci sera affinée par une méthode de relaxation. C'est une méthode qui produit des images plus ou moins convaincantes, mais qui reste limitée à des tissus rectangulaires suspendus à des points d'attache. En 1990, Hinds et McCartney décrivent un logiciel interactif de création de vêtements, où les différentes pièces constituant le vêtement sont définies par un certain nombre de points caractéristiques de leur contour et sont positionnées sur une représentation d'un mannequin. Des plis sont produits à la surface du tissu en utilisant des fonctions sinusoïdales, mais il n'existait aucun contrôle de collision entre les morceaux de tissu et entre le tissu et le mannequin [33]. J. M. Nourrit s'est proposé de modéliser des textiles à base de mailles. Pour modéliser un liage, il a utilisé des courbes Splines qui déterminent la trajectoire du fil et dont les points de contrôle sont ajustés en fonction de règles dépendant des paramètres du liage. Le modèle proposé par l'auteur permet de simuler la plupart des liages réalisables sur des métiers industriels, et peut être utilisé comme outil de conception de nouveaux modèles [13].

3.2 Modèles physiques

Les modèles physiques été une cible des recherches depuis les années 80. En 1987 Terzopoulos et Fleischer, ont caractérisé la simulation de vêtements comme un problème de surfaces déformables et ils ont utilisé la méthode des éléments finis et les techniques de minimisation de l'énergie empruntée à la mécanique [30].

L'intégration implicite et les dérivées optimisées permettent une simulation rapide des propriétés mécaniques du tissu. Toutefois, la vitesse de calcul reste encore lente pour les vêtements complexes, et ces méthodes sont encore limitées par le nombre maximal de polygones, mais peuvent être animées en temps réel [27].

Les modèles physiques se divisent en deux catégories : les modèles continus et les modèles discrets.

3.2.1 Modèles continus

Les modèles continus sont basés sur la théorie de la mécanique des milieux continus. Le milieu, représentant le tissu, est supposé avoir des caractéristiques mécaniques continues. Des relations lient les déformations aux contraintes subies par le milieu. Les déformations représentent l'ensemble des modifications de la géométrie du tissu, par rapport à un état donné (origine des temps, état de repos…). Les contraintes désignent les forces élémentaires s'exerçant en un point du tissu [13]. Il existe essentiellement deux méthodes pour établir le modèle dynamique d'un système en mécanique des milieux continus : le formalisme de **Newton-Euler** [33] et celui de **Lagrange** [33].

En 1986, dans [36], Feynman a implémenté un modèle qui calcule la configuration statique d'un vêtement drapé sur des objets rigides par la gravité. Son modèle se base sur la modélisation du tissu comme une surface déformable élastique continue. Il définit un ensemble de fonctions d'énergie pour le tissu qui sont calculées en représentant le tissu par une grille bidimensionnelle de points tridimensionnels. L'idée essentielle est de calculer la position finale du vêtement par la minimisation des énergies. Dans [37] Terzopoulos et al ont conçu un modèle pour les courbes non rigides, surfaces et solides qui héritent les propriétés essentielles des matériaux élastiques, leur modèle est actif, répondant aux forces externes et interagissant avec d'autres objets. Ils ont utilisé l'équation de Lagrange pour décrire la dynamique d'un corps déformable.

L'équipe de recherche de Thalmann et Magnenat-Thalmann [38,52] ont commencé à adapter le modèle de Terzopoulos en modifiant la représentation de l'amortissement afin de modéliser plus précisément la dissipation de l'énergie au sein du matériau. Ils ont géré les collisions en dehors du modèle élastique en utilisant les lois de conservation des moments, et ce, pour rendre le traitement des collisions plus réaliste. En 1992, Thalmann et al dans [38] ont conçu un modèle pour l'animation de vêtement sur des acteurs synthétiques virtuels en mouvement. En commençant par concevoir les morceaux de vêtement avec des panneaux polygonaux bidimensionnels, puis en les attachant sur le corps de l'acteur virtuel tridimensionnel. Ensuite, des propriétés physiques sont simulées et les vêtements sont animés selon le mouvement de l'acteur dans un environnement physique.

Les modèles continus sont basés sur la théorie de l'élasticité qui considère que le tissu est un milieu homogène, et établissent des lois qui lient les déformations du tissu aux

contraintes qui lui sont appliquées de façon continue dans l'espace et le temps. La loi d'évolution du système au cours du temps est obtenue en passant par une étape de discrétisation du milieu soit par différences finies ou par éléments finis [33].

3.2.2 Modèles discrets

Les modèles discrets considèrent, quant à eux, l'objet à décrire comme un ensemble fini de sous-objets plus simples. Les sous-objets sont munis de lois qui régissent leurs interactions internes d'une part, et leurs interactions avec le reste du monde, d'autre part. Ces lois sont souvent issues de la physique et d'un assemblage de sous-objets appelés systèmes de particules [33]. Il existe deux types principaux de modèles discrets qui ont été développés pour la simulation de vêtements.

a) Modèle des particules

Le modèle de particules [31] a été introduit par Breen & House en 1992. Ils avaient développé un modèle pour la simulation de tissu drapé à l'aide d'un système de particules en interaction qui représentait la structure mécanique sous-jacente du tissu. Les particules interagissent avec leurs particules adjacentes ainsi qu'avec l'environnement qui les entoure en utilisant les équations décrivant les connexions mécaniques associées, représentées par des fonctions d'énergie. Suite à cela, une technique de descente de gradient stochastique a été utilisée comme étant une méthode de relaxation pour mettre les particules du tissu dans un état de repos stable. En 2003, Jakobsen présenta une méthode rapide particulièrement bien adaptée pour les implémentations basées GPU. Il a utilisé le schéma d'intégration de Verlet qui ne nécessite pas une vitesse explicitement donnée. Seules les positions antérieures et actuelles sont nécessaires [13].

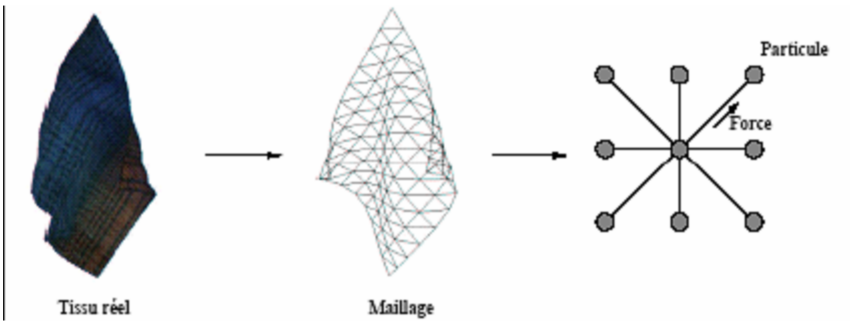


Figure II.1 Discrétisation d'un textile en un maillage polygonale [39].

La figure II.1 présente une discrétisation d'un textile en un maillage polygonale, les sommets sont appelés particules, sa topologie définit les interactions et les forces entre les particules. Le voisinage de chaque particule constituant le tissu est extrait du maillage. À partir de cette information, il est possible de calculer les forces exercées sur chaque particule, celles-ci sont calculées à partir de la position et de la vitesse de la particule considérée ainsi que les positions et vitesses des particules voisinent [39].

b) Modèle masse-ressort

Ce modèle a été initialement introduit en 1988 par Hamann et Parent, puis développé davantage par Provot en 1995 [31], qui a été considéré comme le système masse-ressort le plus utilisé pour la simulation dynamique de vêtements. Il consiste à modéliser une pièce de tissu par une structure déformable constituée d'un réseau contenant des masses ponctuelles liées par des ressorts. Le mouvement du tissu est évalué grâce à l'intégration numérique de la loi fondamentale de la dynamique. Il a décrit les inconvénients de la propriété d'élasticité de ce modèle et a montré que ce problème ne peut pas être résolu par l'augmentation de la raideur des ressorts déformés du fait que celle-ci fait croître le coût de l'algorithme. Il a ainsi proposé une méthode inspirée de la loi de la dynamique inverse pour limiter les trop grands étirements dans le tissu en imposant une distance maximale d'élongation du ressort [13].

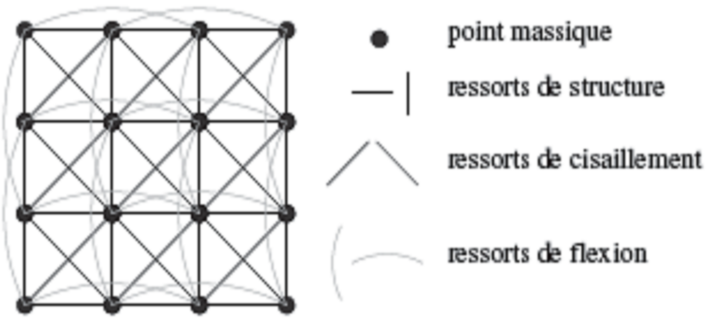


Figure II.2 Le système masse-ressort de Provot [33].

La figure II.2 représente le système masse-ressort de Provot qui est constitué de trois principaux types de ressorts qui se rapportent aux caractéristiques du tissu :

- **Ressorts structurels**: produisent l'allongement et la compression et sont reliés verticalement et horizontalement.
- **Ressorts de cisaillement**: manipulent les contraintes de cisaillement et sont connectés en diagonale.

- **Ressorts de flexion**: s'occupent des flexions et sont reliés verticalement et horizontalement à toutes les autres particules.

Les ressorts sont les éléments structurels du modèle, ils résistent aux différentes charges qui sont appliquées aux particules. Lorsque la simulation est lancée, la longueur au repos de chaque ressort est réglée à la longueur initiale du ressort. Les valeurs de masse des particules et les constantes des ressorts sont définies par l'utilisateur. Une fois les ressorts et les particules mis en place, la simulation est progressivement avancée par l'intégration de la seconde loi du mouvement de Newton. Quand une force de l'environnement, comme la gravité, par exemple, est appliquée aux particules dans le modèle, sur un pas de temps spécifié, elle produit une accélération qui en résulte pour chaque particule. Cette accélération donne lieu à une vitesse qui incite la particule à mettre à jour sa position. La nouvelle position de chaque particule à son tour, entraîne une modification de longueur pour chaque ressort connecté [31].

Le modèle de Provot a été utilisé par la suite comme une base pour les travaux qui ont visé à améliorer d'autres aspects de la simulation de vêtements tels que la précision et le temps de calcul [33]. Parmi ces travaux, celui de Louchet qui a proposé dans [40] un modèle appliquant les techniques de la génétique évolutionnaire pour l'identification des paramètres internes d'un modèle d'animation masse-ressort. Nocent [95] a proposé, de son côté, une amélioration en appliquant un pas de temps adaptatif. Tandis que Baraff a proposé dans[41] un modèle décrivant un système de simulation de vêtements, qui peut et d'une manière stable, prendre de larges pas de temps en couplant une nouvelle technique pour imposer des contraintes sur les particules individuelles du vêtement avec une méthode d'intégration implicite. Dans [42], Volino et al ont proposé un modèle basé sur une méthode optimisée par le calcul des forces élastiques entre les sommets d'un maillage triangulaire irrégulier. La méthode combine la précision de la modélisation de l'élasticité et la vitesse du système masse-ressort simple, en augmentant le pas de temps et en maintenant la stabilité utile pour les applications interactives. Desbrun, quant à lui, a amélioré le schéma d'intégration implicite en proposant un schéma d'intégration permettant d'obtenir une animation réaliste et interactive de n'importe qu'elle système masse-ressort [43]. Dans [45], Burgy a proposé une méthode d'intégration de la multi résolution dans un système de déformation masse-ressort. Choi et al ont poursuivi l'évolution de cet axe en proposant une technique stable de simulation de tissu semi-implicite avec la simulation de plis [44]. C'est une technique qui permet de traiter l'instabilité due au froissement du tissu sans introduire une force d'amortissement dans

la simulation dynamique. Fuhrman et son équipe ont proposé le remplacement des forces internes du tissu par des contraintes qui peuvent être paramétrées pour générer différents comportements du vêtement simulé [24]. Florence Zara dans son travail [39] a proposé une méthode d'optimisation en utilisant les algorithmes parallèles de la simulation physique à l'aide de l'environnement ATHAPASCAN, et ce, pour avoir une simulation de textile en parallèle qui peut être visualisée sur plusieurs écrans. Benameur et Djedi, dans leur travail [46], ont proposé une méthode d'optimisation intégrant la multirésolution basée sur la subdivision des triangles du maillage permettant l'annulation de la subdivision dans les zones raffinées et suffisamment plates avec basculement dans les deux sens.

3.3 Approches hybrides

Les approches hybrides visent à combiner correctement les modèles physiques et les modèles géométriques. L'idée de l'approche hybride résulte des observations suivantes : les simulations physiques sont lentes à calculer, mais elles produisent des résultats réalistes tandis que les simulations géométriques sont beaucoup plus rapides, mais ne sont pas vraiment appropriées pour animer tous les vêtements. En combinant les avantages des deux approches, on peut s'attendre à des résultats acceptables dans des temps de calcul modérés. Dans la plupart des cas, les modèles physiques sont utilisés pour calculer les mouvements globaux de vêtements et les détails tels que les plis sont générés avec des modèles géométriques [27]. Plusieurs travaux ont visé à combiner les deux approches physique et géométrique tels que ceux de Kang [49] qui a présenté dans son travail, une méthode d'animation efficace d'un tissu plissé avec une intégration implicite. En implémentant la méthode, les auteurs ont constaté que l'augmentation de son efficacité est liée à la réduction du nombre de masses du système masse-ressort utilisé.

Cependant, et puisque cette réduction influe sur le réalisme de la simulation, ils ont introduit des courbes Splines cubiques pliées qui génèrent des courbes similaires aux courbes splines cubiques avec des plis selon la distance entre deux points de contrôle adjacents. Dans [47], Hadap et al ont proposé de simuler les plis d'un vêtement avec un plan de relief (bumpmap). Le mouvement global du maillage est simulé avec un système de particules. Le plan de relief / déplacement est le produit du plan de modulation et un schéma de plis. Le plan de modulation est généré par le calcul de déformation locale des triangles du tissu.

3.4 Approche basée sur les exemples

L'approche basée sur des les exemples, connue aussi sous le nom de déformations dérivées par les données, est devenue très populaire depuis plusieurs années. Elle a été utilisée avec succès pour les données de mouvement et d'autres objets graphiques tels que les visages humains et les modèles de corps déformables. L'idée principale de cette méthode est de construire un simulateur qui peut apprendre les déformations à travers une série d'exemples. Ce simulateur peut être considéré comme une boîte noire de calcul telle que pour chaque paramètre d'entrée (interaction avec l'utilisateur), correspond une sortie adéquate (forme de l'objet déformable). Au cours de la phase de formation ou d'entrainement, une série d'entrées / sortie significatives est donnée au simulateur, pour les apprendre. Au cours de l'exécution, le simulateur, lors de la réception d'une entrée, calcule la sortie correspondante. Si l'entrée donnée correspond exactement à l'un des échantillons, le simulateur retournera la sortie correspondante de l'échantillon. Sinon, la chaîne retournée sera interpolée en utilisant une des techniques d'interpolation telles que RBF, K-plus proches voisins, les réseaux de neurones, ou l'interpolation linéaires, etc. James, en 2003, a prouvé que l'approche basée sur les exemples peut être utilisée dans le domaine de la simulation de vêtements. Les exemples de simulation de tissu qu'ils ont utilisés ont été calculés à l'aide d'un logiciel graphique commercial. Leur approche utilise les fonctions de réponse pulsionnelle (IRF). Chaque IRF contient une séquence d'état qui représente une orbite d'un sommet du tissu. Au cours de l'exécution de la simulation, l'orbite correspondant à l'interaction de l'utilisateur est jouée. Si l'interaction de l'utilisateur est modifiée, une nouvelle orbite qui correspond à la nouvelle interaction est trouvée, remplaçant l'actuelle. Cette méthode n'est pas adaptable à la simulation de vêtements sur des humains virtuels [48].

4. Conclusion

La simulation de vêtements a une grande importance dans l'animation par ordinateur. Les recherches dans ce domaine sont en plein essor, visant à atteindre un niveau de réalisme supérieur tout en respectant un temps de calcul réel.

Nous avons donné au cours de ce chapitre un aperçu sur la simulation de vêtements expliquant ses différentes méthodes et techniques appliquées.

Chapitre III

Gestion de collisions : Application à la simulation de vêtements

I. Introduction

Le domaine de la simulation des vêtements virtuels évolue en impliquant une large combinaison de techniques qui relèvent beaucoup de défis concernant le réalisme (aspect qualitatif) et le temps réel (aspect quantitatif). Produire des simulations de vêtements réalistes, revient à produire exactement et d'une manière objective le comportement réel du vêtement. Ce qui est bien loin d'être satisfait et qui implique le la tolérance de la justesse physique et visuelle. Chercher le réalisme dans une simulation de vêtements nous oblige à nous intéresser à la gestion des collisions [33]. La gestion des collisions dans la simulation de vêtements, qui sont considérés comme des objets déformables, est nettement plus compliquée que celle des objets rigides. Contrairement aux objets rigides, un morceau de tissu a tendance à entrer en collision avec lui-même. Pour avoir une simulation correcte, deux types de collisions doivent être détectées, la collision : vêtement- environnement et la collision : vêtement- vêtement.

Dans ce qui suit, nous allons présenter la gestion de collision dans la simulation de vêtements, en commençant par expliquer la méthode basique de la détection de collision, suivie des différentes techniques accélératrices qui visent à améliorer la performance et la qualité de la simulation. Nous allons ensuite aborder le traitement de collision et ces différentes méthodes, puis nous donnerons une idée générale sur les auto-collisions et les collisions continues, suivie d'une analyse englobant quelques travaux qui gérant les collisions et qui rentrent principalement dans le domaine de la simulation des vêtements. Le chapitre sera clôturé par un bref aperçu sur la multi résolution ainsi qu'une conclusion.

II. Gestion de collisions en animation de vêtements

Dans l'animation des objets déformables, la gestion de collisions est le principal goulot d'étranglement. La détection et la réponse à la collision sont des points cruciaux pour la performance des applications de l'animation des objets déformables. Contrairement aux objets volumétriques, les exigences pour la précision de la gestion des collisions des textiles sont particulièrement plus strictes parce que tout chevauchement est visible. Par conséquent, il est utile d'appliquer des méthodes spécialement conçues pour les surfaces déformables qui accélèrent la détection de collisions.

Les techniques de détection de collisions reposent sur la représentation de l'objet à détecter. Une solution courante et commune consiste à représenter à la fois le tissu et l'environnement (la peau humaine, chaises ...) sous forme de maillages triangulaires. Dans ce cas, la détection de collisions est généralement réalisée en cherchant les collisions entre les triangles et les sommets en plus des collisions arête/arête entre deux maillages en mouvement.

Afin d'éviter des comparaisons l'ordre de $O(n^2)$ (où n est le nombre de triangles), des structures d'accélération comme les volumes englobants sont utilisés. Les volumes englobants les plus utilisés sont les sphères englobantes, les boîtes englobantes (AABB) et les boîtes englobantes orientées (OBB). Les volumes englobants peuvent être organisés de façon hiérarchique afin de réduire davantage le calcul. Pour le modèle des tissus, le partitionnement binaire de l'espace (BSP) qui est un arbre obtenu par partitionnement récursif des maillages du tissu en fonction de leurs coordonnées est le plus utilisé [26]. Dans ce qui suit nous allons discuter sur le fait que la gestion de collisions pour les vêtements, qui sont des objets déformables, soit plus compliquée et plus difficile que la gestion de collision des objets rigides, et cela en citant quelques aspects compliquant cette tâche.

1. Détection de collisions pour la simulation de : vêtements VS objets rigides

Le tissu est un matériau extrêmement souple et montre une très faible résistance à la flexion, il y a différents aspects qui compliquent le problème de détection de collisions pour les vêtements par rapport aux corps rigides, parmi lesquels nous pouvons citer [56]:

- Afin de simuler de façon réaliste des interactions entre objets déformables, tous les points de contact, y compris ceux qui sont dus à l'auto-collision doivent être considérés. Par contre, pour les objets rigides, l'auto collision ne se produit pas souvent et si elle se produit, elle est souvent négligeable. Ainsi, ce point complique

de manière significative la gestion de collisions en ajoutant le module d'auto-collision.

- Les algorithmes de détection de collisions efficaces utilisent parfois des techniques d'accélération basées sur les structures de données spatiales comme les volumes englobants ou les champs de distances. Ces représentations d'objets sont généralement établies dans une étape de prétraitement et s'exécutent très bien pour les objets rigides, car le temps passé dans cette étape peut être négligé et les structures de données spatiales peuvent être réutilisées pour d'autres étapes de simulation. Néanmoins, s'agissant des objets, déformables, il est nécessaire de mettre à jour ou reconstruire ces structures de données après chaque étape de déformation, ce qui a principalement comme conséquence une efficacité globale réduite.
- Le traitement de collisions pour les objets déformables est très complexe et en particulier pour les objets surfaciques comme le tissu où des objets peuvent complètement le traverser. Par conséquent, pour rapporter une réponse stable, les algorithmes de détection de collisions doivent fournir des informations détaillées telles que la profondeur de pénétration ou la direction des objets en collision.
- Il y a un grand nombre de collisions possibles comme dans les maillages déformables. Potentiellement, chaque sommet peut entrer en collision avec n'importe quel triangle et chaque arête peut entrer en collision avec n'importe quelle autre arête. Ce qui implique que le nombre d'éléments qui peuvent entrer en collision est notamment plus élevé pour les objets déformables que pour les objets rigides.

2. Détection de collision basique

Le cas général de manipulation de collisions est celui impliquant un objet en tissu et un autre objet mobile dans la scène. Un cas particulier est le cas des auto-collisions, les deux cas sont généralement traités de manières similaires, et ce, en considérant principalement les vêtements et les objets comme des maillages triangulaires.

Soit t_0 un instant où il n'y a pas d'interpénétration entre le vêtement et l'objet. Considérons l'intervalle de temps $[t_0, t_0+\Delta t]$, connaissant les positions et les vitesses de chaque sommet de notre modèle au temps t_0, il est possible de calculer sa position au temps $t_0+\Delta t$. La

détection de collisions consiste alors à trouver si une ou plusieurs collisions se sont produites pendant cet intervalle de temps. Ces collisions peuvent être de deux types :

- Soit un sommet de l'un des deux maillages est entré en collision avec un triangle de l'autre maillage (collision sommet/triangle).
- Soit une arête d'un triangle de l'un des deux maillages entrée en collision avec une autre arête d'un triangle de l'autre maillage (collision arête/arête).

Pendant le processus de simulation de vêtements, il est utile d'utiliser une méthode d'intégration numérique telle que la méthode explicite d'Euler, pour garder une vitesse constante du sommet durant son mouvement pendant l'intervalle de temps [t_0, t_0+Δt]. Commençons par expliciter le premier type de collisions.

2.1 Collision sommet/triangle

Soit P (t) le point (sommet) en mouvement, et A(t), B(t), C(t) les sommets du triangle en mouvement. Soit $\overrightarrow{V}$,$\overrightarrow{V_A}$,$\overrightarrow{V_B}$,$\overrightarrow{V_C}$ leurs vitesses respectives durant l'intervalle de temps [t_0, t_0+Δt]. Donc on aura :

$A(t) = A(t_0) + t\overrightarrow{V_A}$

$B(t) = B(t_0) + t\overrightarrow{V_B}$

$C(t) = C(t_0) + t\overrightarrow{V_C}$

S'il y a une collision, alors le point P(t) appartiendra au triangle ABC(t), ce qui peut être exprimé comme suit :

$\exists t \in [t_0, t_0+\Delta t]$ tel que $\exists$ u, v $\in$ [0,1], u + v$\leq$ 1, $\overrightarrow{AP}(t) = u\overrightarrow{AB} + v\overrightarrow{AC}$

Cependant, cette équation vectorielle nécessite un système d'équations non linéaire, dont la résolution exige l'introduction et la prise en compte d'une autre condition qui prouve que P appartient au triangle ABC. En effet, à partir du produit vectoriel ($\overrightarrow{AP}(t) . \overrightarrow{N}(t) = 0$), on produit une équation de troisième degré, donc trois valeurs de t peuvent être obtenues et uniquement celles qui appartiennent à l'intervalle [t_0, t_0+Δt] correspondent à une collision. En les injectant dans le premier système d'équations, celui-ci deviendra linéaire. Si plusieurs valeurs de (t,u,v) sont des solutions au système, il faut choisir la valeur la plus petite de t [58].

2.2 La collision arête/arête

Ce type concerne la détection de collisions durant un intervalle de temps [t_0, t_0+Δt], entre une arête du vêtement et une arête de l'objet en mouvement.

Soit AB(t) la première arête et CD(t) l'autre arête, il y aura collision si et seulement si :

$\exists t \in [t_0, t_0+\Delta t]$ tel que

$\exists u, v \in [0,1]$, $u\,\overrightarrow{AB}(t) = v\,\overrightarrow{CD}(t)$

Ceci nous ramène à un système non linéaire, une autre relation sera utilisée pour trouver la valeur de t en évitant de résoudre le système. Les quatre points A, B, C, D, s'alignent sur le même plan, ainsi:

$(\overrightarrow{AB}(t) \wedge \overrightarrow{CD}(t)).\overrightarrow{AC}(t) = 0$

Cette relation produit une équation du troisième degré et permet de calculer u et v en injectant t dans la première équation, pour détecter une collision, si elle existe [58].

Dans ce qui suit, nous allons présenter quelques techniques de détection de collisions qui ont été utilisées dans différentes recherches sur la simulation de vêtements et plus généralement la simulation des objets déformables.

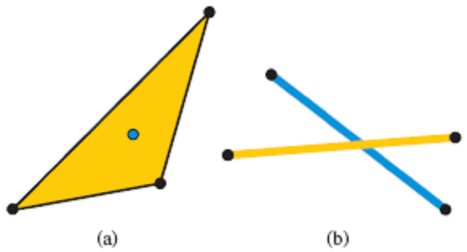


Figure III.1 Différentes configurations de collisions:

(a) les collisions sommet-triangle, (b) les collisions arête-arête [60].

3. Techniques de gestion de collision

Plusieurs approches ont été introduites dans la résolution de détection de collisions, ces techniques visent plus particulièrement à accélérer cette détection. Nous allons d'abord présenter quelques approches puis nous essaierons de discuter les caractéristiques de chacune de ces approches. Commençons par la méthode de la hiérarchie des volumes englobants.

3.1 La hiérarchie des volumes englobants

L'idée de la hiérarchie des volumes englobant**s** (BVHs) est de diviser chaque objet en un ensemble de primitives contenues dans des volumes englobant**s** (BVs). Les BVHs accélèrent la détection de collision**s** de deux manières: d'abord, une détection de collision**s** entre BVs est beaucoup plus rapide qu'un contrôle de collision**s** entre les objets inclus dans

les BVs. En second lieu, la hiérarchie est employée pour éviter efficacement les tests de collisions inutiles dans les régions où la collision ne se produit pas [60]. En général, les BVHs sont définis comme suit: chaque nœud dans l'arbre est associé à un sous-ensemble de primitives de l'objet, avec des volumes englobants qui entourent les sous-ensembles avec la plus petite instance de contenants qui appartiennent à une classe de formes spécifiques. Les volumes englobants qui ont été employés pour la méthode des BVHs sont des sphères, des boîtes englobantes orientées (OBBs), des polytopes orientés discrets (DOPs), les Boxtrees, les axis aligned bounding boxes(AABBs), et les enveloppes convexes.

Bien qu'on ait proposé une variété de BVs, deux types méritent d'être: mentionnés: les OBBs et les k-DOPs. En notant que les AABBs sont un cas spécial des k-DOPs avec k = 6.

Les OBBs ont la propriété que, dans certaines suppositions, permet à leur raideur d'augmenter linéairement pendant que le nombre de polygones diminue. Les k-DOPs, d'autre part, peuvent se rapprocher arbitrairement de l'enveloppe convexe en augmentant k. De plus, les k-DOPs, particulièrement ceux avec k = 6, peuvent être calculés très efficacement. C'est important, pour les objets déformables qui exigent des mises à jour fréquentes de la hiérarchie [56].

La figure III.2 présente les différents volumes englobants qui peuvent être utilisés dans la hiérarchie des volumes englobants.

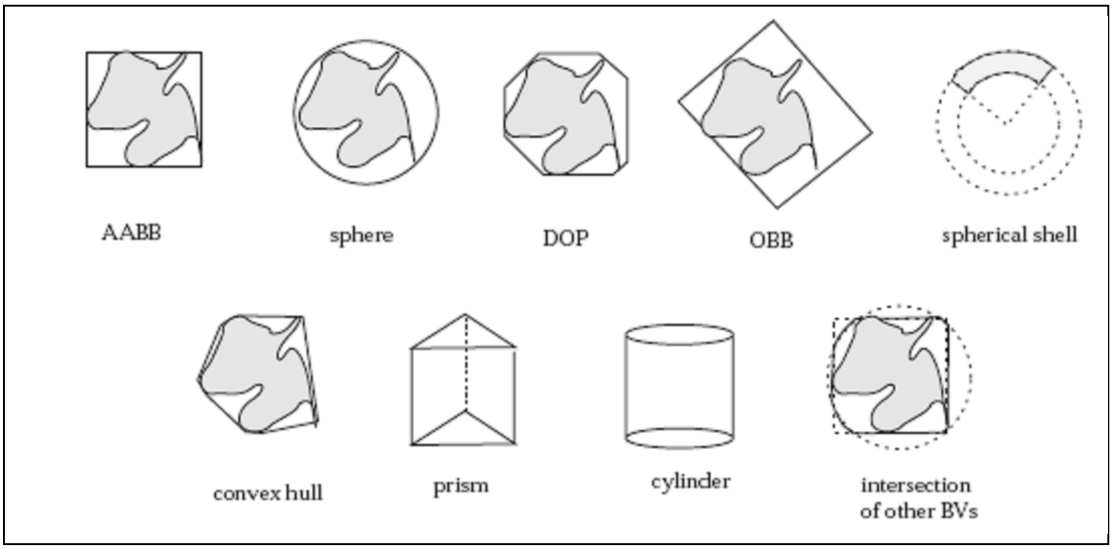


Figure III.2 Une variété de volumes englobants proposés pour la détection de collisions basée sur la méthode de la hiérarchie des volumes englobants BVHs [56].

La figure III.3 illustre un simple exemple d'une hiérarchie de sphères englobantes avec des éléments d'un maillage triangulaire.

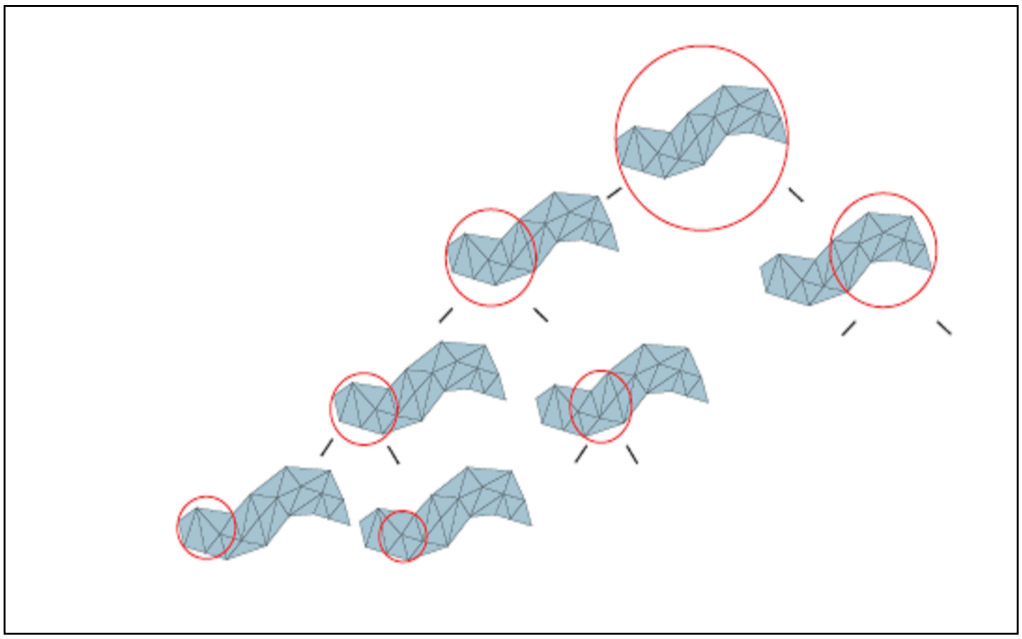


Figure III.3 Quatre niveaux d'un arbre binaire de sphères englobantes sur un maillage triangulaire [60]

En utilisant ces BVs, dans une étape de prétraitement avant l'étape réelle de détection, pour chaque objet pouvant être en collision, une BVH est construite. L'objet est partitionné jusqu'à ce qu'un certain critère de feuille dans la hiérarchie soit rencontré. Le plus souvent, chaque feuille contient une primitive simple, mais on pourrait aussi bien s'arrêter quand un nœud contient moins qu'un nombre de primitives spécifié. Ici, les primitives sont des entités qui composent les objets graphiques, qui peuvent être des polygones ou des polyèdres, etc. [60].

3.1.1 Parcours de la hiérarchie

Pour le test de collisions de deux objets ou le test d'auto-collision d'un seul objet, les BVHs sont traversés de haut en bas et des paires de nœuds de l'arbre sont récursivement testées s'il y a chevauchement. Si les nœuds chevauchés sont des feuilles alors les primitives inclues sont testées pour l'intersection. Si un nœud est une feuille tandis que l'autre est un nœud interne, le nœud de feuille est examiné contre chacun des enfants du nœud interne. Si, cependant, tous les deux nœuds sont des nœuds internes, la probabilité de l'intersection est réduite au minimum aussi rapidement que possible. Par conséquent, on

doit tester le nœud avec le plus petit volume contre les enfants du nœud avec le plus grand volume, ceci s'illustre dans la figure III.4 [56].

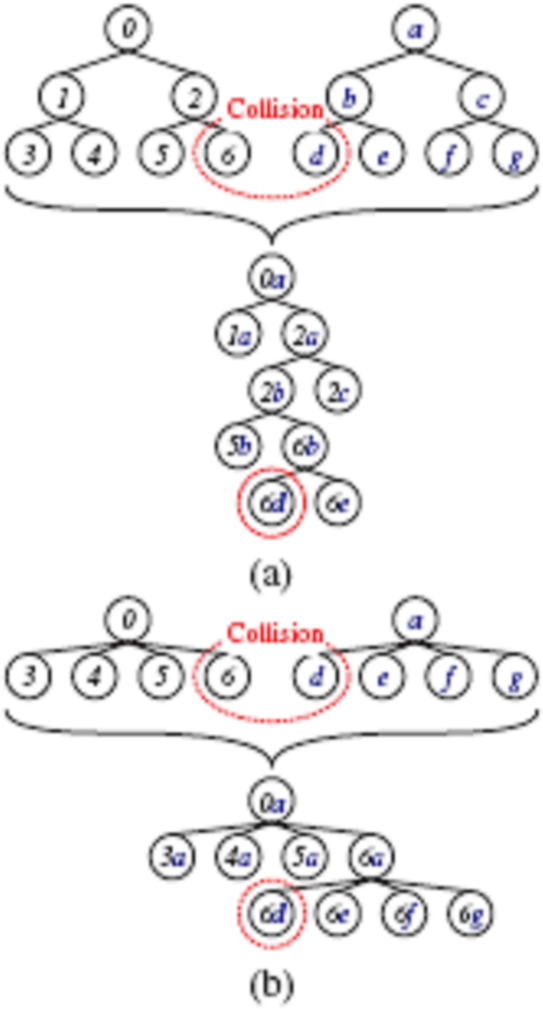


Figure III.4 Récursivité en utilisant :(a) arbres binaires (b) arbres 4-aire [56].

Pour deux objets donnés avec les BVHs A et B, la plupart des algorithmes de détection de collisions mettent en application le schéma général de l'algorithme suivant[60]:

```
Traverse(A,B)
If A and B do not overlap then
return
end if
if A and B are leaves then
returnintersection of primitives enclosed by A and B
else
for all children A[i] and B do
traverse(A[i],B)
end for
```

```
end if
```

Les caractéristiques des différents algorithmes hiérarchiques de détection de collisions reposent sur le type de BVs utilisés, le test de chevauchement d'une paire de nœuds, et l'algorithme de construction des arbres de BVs [56].

3.1.2 Construction de la hiérarchie

Il existe trois stratégies différentes pour établir des BVHs, à savoir : descendante (top-down), ascendante (bottom-up), et la stratégie d'insertion. Cependant, la stratégie descendante est généralement la plus utilisée pour la détection de collisions. L'idée est de subdiviser périodiquement un ensemble de primitives d'objets jusqu'à ce qu'un seuil soit atteint. Cette subdivision est guidée par un critère spécifié par l'utilisateur ou une heuristique qui produira une BVH respectant le critère choisi. Une heuristique de subdivision très simple consiste à approximer chaque polygone par son centre. Puis, pour un ensemble B de points donnés, à calculer ses composants principaux (les vecteurs propres de la matrice de covariance); choisir le plus grand entre eux (c.-à-d., celui montrant la plus grande variance); placer un plan orthogonal en cet axe principal et par le barycentre de tous les points de B. Ceci subdivise B en deux sous-ensembles. Alternativement, le plan de subdivision peut être placé sur la médiane de tous les points. Ceci mène à un arbre équilibré. Cependant, il est peu clair que les arbres équilibrés fournissent une efficacité améliorée pour la détection de collisions [56].

3.1.3 La mise à jour de la hiérarchie

Il existe différentes stratégies, non seulement pour établir une hiérarchie, mais également pour la mise à jour de la hiérarchie. Larson et al, dans leurs travaux [78] ont comparé les stratégies top-down et bottom-up, ce qui leur a permis constater que si dans un procédé de détection de collisions beaucoup de nœuds profonds sont atteints, la stratégie ascendante s'exécute mieux, alors que si seulement quelques nœuds profonds sont atteints, l'approche descendante s'avère plus rapide.

Par conséquent, ils ont proposé une méthode hybride qui met à jour la moitié supérieure de l'arbre avec l'approche ascendante et ont utilisé la stratégie descendante seulement s'il reste des nœuds atteints non mis à jour. En utilisant cette méthode, ils réduisirent le nombre de nœuds inutilement mis à jour avec l'inconvénient des besoins en mémoire plus élevés parce qu'ils doivent stocker les informations des feuilles sur des sommets et les faces ainsi qu'au niveau des nœuds internes.

Mezger et al ont proposé d'autres approches [62] pour pouvoir accélérer la mise à jour de la hiérarchie en omettant ou en simplifiant le processus de mise à jour pour plusieurs

étapes de temps. Pour ce faire, les volumes englobants peuvent généralement être gonflés avec une certaine distance. Ainsi, la mise à jour de la hiérarchie n'est pas nécessaire tant que les primitives englobées ne se déplacent pas plus loin que cette distance.

3.1.4Présentation de quelques volumes englobants

Les volumes englobants sont de types différents et chacun d'entre eux a ses points forts et ses points faibles. Il existe plusieurs volumes englobants dont nous allons présenter les plus utilisés dans le domaine de la détection de collisions des objets déformables en général, nous pouvons citer: [64]

- **Axis-Oriented Bounding Boxes (AABB)**

Ce sont des boîtes rectangulaires avec six surfaces et ses normales de faces sont parallèles avec les axes du système de coordonnées (voir la figure III.5). D'une façon générale, les AABB peuvent être représentés en tant que valeur minimale et valeur maximale par rapport à l'axe des coordonnées des objets englobés et ils sont très rapides et efficaces, mais pas très bien appropriés.

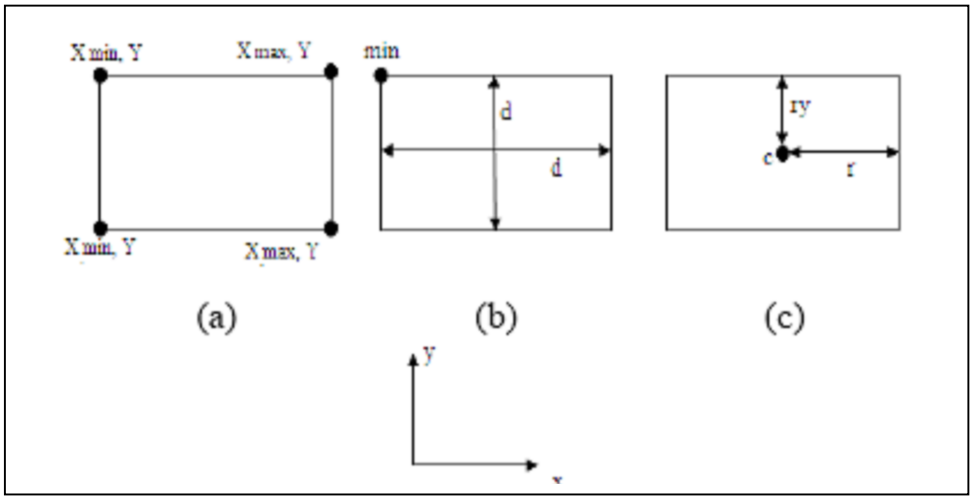


Figure III.5 Représentation des AABB : (a) min-max, (b) min-largeur, (c) centre-rayon [64].

- **Les sphères englobantes**

Fondamentalement, les valeurs les plus importantes des sphères sont le centre et le rayon. Ainsi, les sphères englobantes représentent, la plupart du temps, le choix le plus efficace parce que le test d'intersection entre les sphères englobantes est très rapide (comparant la distance entre les centres des sphères) et simple comparé aux AABBs et à tout autre volume englobant, la figure III.6 représente ce type de volume englobant.

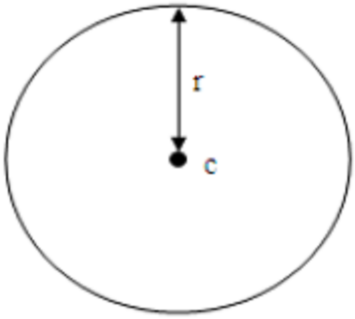


Figure III.6 Représentation d'une sphère englobante [64].

- **Les k-DOPs**

Un k-DOP(polytope orienté discret) est un polyèdre convexe défini par k demi-espaces, notés comme suit :

$H_i = \{x \in R^m | n_i^T x \leq b, n_i \in N, b_i \in R\}$

Les normales n_i des hyperplans correspondants de tous les k-DOPs sont discrétisés et forment le petit ensemble $N= \{n_1, \ldots, n_k\} \subset R^m$. Pour des raisons arithmétiques, les composants des vecteurs des normales sont habituellement choisis dans l'ensemble {-1,0,1}. Afin de transformer les tests d'intersection des polyèdres en de simples tests d'intervalles, les hyperplans doivent former k/2 paires parallèles. Exemple : les AABBs sont des 6-DOPs dans R^3, ils sont donnés par $N_6= \{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\}$.

La façon la plus simple des volumes englobants de k-DOP pour un ensemble de points consiste à les insérer dans un k-DOP primaire en mettant à jour ces k/2 intervalles. Le test d'intersection entre deux k-DOPs est mis en application par des tests d'intervalles semblables à l'AABB commun, indiquant les détachements comme une paire d'intervalles disjointe. Ainsi, le nombre maximal de tests d'intervalles est k/2 (dans le cas de recouvrements) [62].

3.1.5 Détection d'auto-collision avec la hiérarchie des volumes englobants

Les BVHs peuvent être facilement utilisés pour accélérer les auto-collisions, qui sont particulièrement importantes pour les objets déformables (vêtements). En général, les collisions et les auto-collisions sont exécutées par le même algorithme en utilisant les BVHs. Si plusieurs objets sont testés pour des collisions, les BVHs respectifs sont vérifiés les uns contre les autres. De façon analogue, les auto-collisions d'un objet sont détectées en testant la BVH avec elle-même. Cependant, il doit être noté que les volumes englobants (BVs) des régions voisines peuvent se chevaucher ou entrer en collision, quoiqu'il n'y ait aucune auto-collision. Pour éliminer efficacement de tels cas, différentes heuristiques ont

été proposées. Volino et Magnenat-Thalmann[15] ont proposé une méthode exacte pour éviter les tests inutiles d'auto-collisions entre certains BVs. L'idée est basée sur le fait que les régions reliées avec suffisamment moins de courbure ne peuvent pas s'auto-intersecter, en supposant que leur contour est convexe. Par conséquent, un vecteur du produit scalaire positif avec toutes les normales de la région est recherché. Si un tel vecteur existe et la projection de la région sur un plan dans la direction du vecteur ne s'auto-intersecte pas, la région ne peut pas s'auto-intersecter. Une approche très semblable est employée par Provot[58] qui employa des normales en cônes représentant un ensemble des directions de la normale, qui sont calculées pour chaque région convexe. L'angle du cône α représente la courbure indiquant de possibles intersections si $\alpha \geq \pi$[60].

3.2 Champs de distance

Une autre méthode de détection de collisions largement répandue pour la détection de collisions entre le tissu et les objets rigides est les champs de distance. À l'origine, les champs de distances n'ont pas été utilisés pour la détection de collisions, mais ont été employés en infographie pour le rendu des volumes, et pour créer des surfaces de décalage. Un champ de distance est un champ scalaire qui indique la distance minimum à une surface (figure III.7). Si la surface est fermée, la distance peut être signée afin de distinguer l'intérieur et l'extérieur de la forme. La détection de collisions entre les points et l'objet est effectuée en évaluant la distance signée. La représentation d'une surface fermée par un champ de distance présente l'avantage de ne nécessiter aucune restriction sur la topologie. De plus, l'évaluation décrite des distances et l'évaluation des normales nécessaires pour le processus de détection et de traitement de collisions est très rapide et indépendante de la complexité de l'objet [27].

3.2.1Représentation des champs de distances

Pour représenter les champs de distances, différentes structures de données pour le partitionnement de l'espace, comme les grilles 3D uniformes, les arbres Octrees ainsi que le partitionnement binaire de l'espace (arbres BSP) ont été proposés. La représentation des champs de distances la plus simple est celle des grilles uniformes, où les valeurs de distances sont calculées pour chaque point de la grille, tel que les valeurs peuvent être calculées par interpolation trilinéaire. Les avantages principaux de ces grilles uniformes sont la simplicité de mise en œuvre et le temps constant nécessaire pour le calcul des distances à n'importe quel point de la grille. Les normales nécessaires de la surface pour la

plupart des méthodes de traitement de collision peuvent facilement être calculées depuis le gradient analytique de l'interpolation trilinéaire. À côté de ces avantages, cependant, les grilles uniformes ont des inconvénients, résultant de leur taux constant d'échantillonnage. Une possibilité pour surmonter ces problèmes est l'utilisation des champs de distances échantillonnées de manière adaptative (ADFs), là où un échantillonnage dirigé par détails est employé. Des taux élevés d'échantillonnages sont utilisés dans les régions quand le champ de distance contient le détail fin, et un bas taux d'échantillonnage quand les champs varient doucement. Cette stratégie d'échantillonnage la entraîne des précisions arbitraires dans la reconstruction des champs avec un usage efficace de la mémoire.

L'utilisation des arbres BSP en tant que représentation de champs de distances, permet la réduction de la consommation de mémoire, en employant une approximation linéaire par morceaux du champ de distances, qui n'est pas nécessairement continu. L'inconvénient de la représentation par les arbres BSP c'est qu'ils ont des coûts élevés pour leur construction. De plus, les problèmes résultants des discontinuités entre les cellules ne peuvent pas être résolus aussi facilement comme c'est le cas pour les ADFs[56].

3.2.2 Détection de collisions avec les champs de distances

En utilisant les champs de distances, le procédé de détection de collisions entre différents objets est effectué de manière ponctuelle. Les sommets d'un objet sont testés par rapport aux champs de distances de l'autre objet et vice versa. Une collision se réalise si $D(p) > 0$.

Dans le cas de la simulation de tissu, seuls les sommets du maillage du vêtement sont testés pour les collisions éventuelles. Comme les collisions sont non seulement provoquées par des sommets interpénétrant l'autre objet, mais également par les bords qui se heurtent, il est nécessaire de décaler les sommets de l'isosurface nulle par une distance prédéfinie ε. Ce décalage dépend de la résolution du maillage (figure III.8).

Pour ce qui est de la partie a de la figure, pendant la résolution des collisions avec des champs de distances, un décalage est nécessaire, car seuls les sommets sont testés par rapport aux champs de distances tandis que des collisions entre les arêtes sont négligées. Sans décalage des sommets, des interpénétrations et des artefacts peuvent se produire pendant la détection de collisions. Pour ce qui est de la partie b, on opère par une introduction d'un décalage ε peut résoudre ce problème.

Fuhrmann et al appliquèrent dans [65] une approche ressemblant à la subdivision, en testant en plus le centre de chaque arête dans le maillage déformable afin d'améliorer la précision de la détection de collisions.

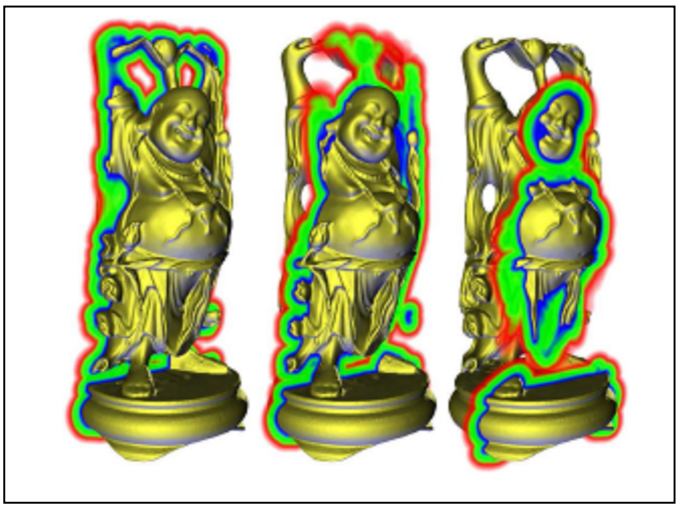


Figure III.7 La statuette de Buddha l'heureux

À titre d'exemple, pour la statuette de Buddha l'heureux (Figure III.7), nous remarquons trois couches de couleurs imprimant le champ de distance. Puisque le champ de distance n'est valide que dans une couche près de la surface, le coloriage est fané en dehors de cette distance. La couche bleue sert à clôturer des distances, tandis que le rouge indique des distances moyennes [27].

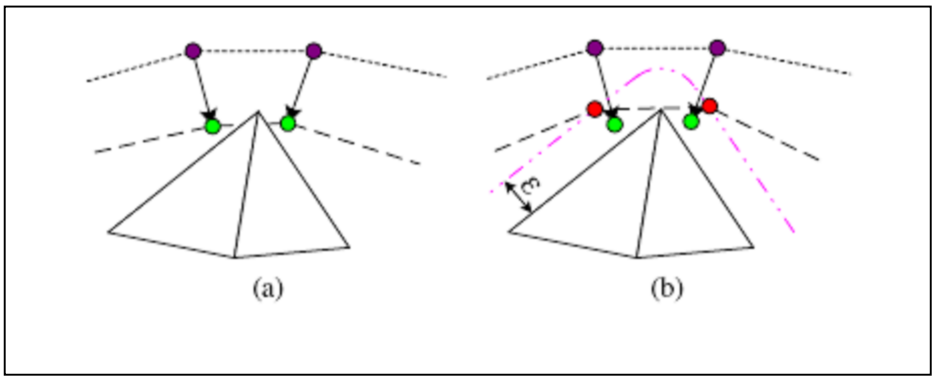


Figure III.8 Application du décalage pendant la résolution des collisions avec des champs de distances [65].

Comme les BVHs, l'application des champs de distances aux corps déformables nécessite une mise à jour de la représentation du champ de distances après chaque déformation, c.-à-d. après chaque pas de temps. La mise à jour des champs de distances constitue également un problème compliqué pour le processus de détection de collisions. Pendant le traitement de collisions, les champs de distance sont déformés et ce à cause de

la géométrie, et sont utilisés pour une approximation de la profondeur de pénétration. Cette méthode est valable pour la détection de collisions comme pour la détection de l'auto-collision des objets volumétriques. Cependant, leur approche n'est bien appropriée que pour les objets volumétriques. Des objets minces comme le tissu ne peuvent pas être représentés par les champs internes de distances et donc aucune détection d'auto-collision vêtement/vêtement n'est possible. Bridson et al [50] utilisèrent les ADFs pour détecter des collisions entre le vêtement et les caractères animés. Ces caractères animés se déforment tout le temps en se basant sur la simulation fondamentale de peau et des muscles. Les champs de distances correspondants pour chaque pas de temps sont précalculés.

Pour calculer les champs de distances et accélérer leurs mises à jour, Vassilev et al [57] ont proposé une approche basée sur l'espace d'image. Ils employèrent le matériel pour la construction de deux cartes de profondeurs et des normales de l'objet, une carte pour le derrière de l'objet et une carte pour le devant. Ces cartes ont été employées pour des calculs de distances et le traitement de collisions. Cependant, cet algorithme véhiculait peu d'exactitude et était restreint aux formes convexes ou à des cartes de directions appropriées dans le cas des caractères animés. L'approche est un mélange entre les champs de distances et les techniques de l'image/espace [60].

3.3 Subdivision spatiale

L'idée principale derrière la subdivision spatiale est que seules les primitives de l'objet sont testées pour la collision et qui, de plus, appartiennent à la même cellule de l'espace partitionné. Par conséquent, les algorithmes de subdivision spatiale procèdent habituellement sur plusieurs étapes. Dans une première étape, les primitives des objets sont assignées aux cellules de l'espace subdivisé. Puis, dans la deuxième étape, on teste quelles sont les primitives des différentes collisions d'objets qui sont dans la même cellule. Si ces primitives s'intersectent, une collision est détectée**.** Si toutes les primitives qui présentent des intersections appartiennent au même objet, une auto- collision est détectée. Comme les méthodes de détection de collisions précédemment présentées, la subdivision spatiale a besoin également d'une étape de mise à jour si les objets se déplacent ou se déforment. L'attribution des primitives aux cellules de la grille doit être renouvelée à chaque pas de temps.

La subdivision spatiale a été d'abord utilisée pour des questions de voisinage. Plus tard, pour la détection de collisions des objets rigides dont diverses approches utilisant différentes représentations pour la division de l'espace ont été présentées. Comme la

subdivision de l'espace utilisée pour les grilles uniformes de champs de distance, les octrees ou les arbres BSP sont dépendants des objets tandis que les subdivisions spatiales utilisant les grilles uniformes sont indépendants des objets et sont par conséquent bien adaptés aux objets déformables respectant le temps de simulation. Une approche combinant le hachage spatial et les grilles uniformes pour la détection des collisions et des auto-collisions des maillages tétraédriques déformables a été utilisée en subdivisant implicitement l'espace R^3 en petites cellules d'une grille et en utilisant une fonction de hachage pour projeter ces cellules 3D de la grille dans une table de hachage. Cette approche est efficace du point de vue de l'usage de la mémoire et fournit une flexibilité, dès lors qu'elle permet de manipuler des grilles spatiales régulières potentiellement infinies. Un autre avantage est qu'aucune structure de données complexe comme des octrees ou BSPs n'est exigée [60].

3.4 Technique de l'espace d'image

Une autre classe de méthodes de détection de collisions est la classe des techniques de l'espace de l'image. Ces algorithmes établissent un rendu sur les objets en les projetant dans diverses directions et en employant par la suite les cartes de profondeurs des images pour trouver les intersections. Pour accélérer le processus de détection, ces projections peuvent être obtenues en utilisant le matériel graphique. Puisque les techniques de l'espace de l'image n'exigent aucun prétraitement, elles sont particulièrement appropriées pour des environnements avec des objets dynamiquement déformables.

Une des approches de détection de collisions par la technique de l'espace d'image est celle proposant d'établir un rendu sur les deux couches de profondeur d'objets convexes (couche avant et arrière) dans deux tampons de profondeurs. L'intervalle entre la valeur de profondeur la plus petite et la plus grande à chaque pixel représente alors approximativement l'objet et est efficacement employé pour la vérification des interférences. D'autres approches ont été présentées, mais elles ont été restreintes aux objets convexes, et ne prenaient pas en considération les auto-collisions. La première application de la technique de l'espace d'image dans la détection de collisions pour la simulation dynamique de tissu a été réalisée par Vassilev et al [57] qui appliquèrent un rendu sur une vue de l'avant et de l'arrière d'un avatar pour produire une carte de profondeur. En utilisant cette carte de profondeur, ils ont pu détecter des sommets du tissu qui s'intersectent. Cependant, leur approche est limitée aux objets convexes, ce qui ne peut assurer la déformation des avatars animés. Par conséquent, la simulation peut

montrer des artefacts dus aux collisions non détectées. Dans des applications médicales, des techniques de l'espace d'image ont été utilisées pour détecter des intersections d'un outil chirurgical avec le tissu de la peau déformable simulée en appliquant un rendu sur l'intérieur de l'outil. Baciu et Wong [67] ont présenté une nouvelle approche de la technique de l'espace d'image qui enrichit l'approche de Vassilev pour supporter des objets non convexes. Ils décomposèrent la surface de l'objet dépendamment de la courbure et appliquèrent le rendu de ces surfaces dans le tampon de profondeur. Cependant, ils n'ont pas exécuté directement le processus de détection de collisions basé sur ces informations, mais ont rassemblé les paires de triangles potentiellement intersectés pour une transformation ultérieure. Ainsi, cette approche consiste en une combinaison entre les techniques de l'espace d'image et la subdivision spatiale. Et en définitive, cette configuration permet de détecter les collisions et les auto-collisions [60].

3.5 Les méthodes stochastiques

Les méthodes inexactes constituent l'axe d'intérêt central du domaine de la détection de collisions. Cet intérêt est motivé par de nombreuses observations. Premièrement, les modèles polygonaux représentent juste une approximation de la géométrie réelle. En second lieu, la qualité perçue de la plupart des applications interactives 3D ne dépend pas de la simulation exacte, mais plutôt de la réponse en temps réel aux collisions. En même temps, les humains ne peuvent pas distinguer entre un comportement physiquement correct et un comportement physiquement plausible des objets. Par conséquent, il peut être toléré d'améliorer la performance de la détection de collisions, tout en dégradant sa précision.

Dans ce qui suit, deux approches de ces méthodes stochastiques vont être présentées. Les deux méthodes emploient des principes probabilistes avec des manières assez différentes. La première emploie des méthodes probabilistes pour estimer la possibilité de collisions avec le respect d'un critère de qualité donné. Avec cette méthode la qualité de la détection de collisions peut être directement spécifiée par l'utilisateur assurant de ce fait plus de contrôle. La deuxième méthode devine initialement les paires en collision à travers un échantillonnage stochastique à l'intérieur des corps en collision. Les régions en collision exactes sont alors rétrécies vers le bas en employant ce principe en conjonction avec la cohérence temporelle et spatiale. Dans ce cas, l'utilisateur a un contrôle plus indirect de la qualité de la détection de collisions.

3.5.1 Approche D'Average-Case

Conceptuellement, l'idée principale de cet algorithme est de considérer des ensembles de polygones aux nœuds intérieurs des BVHs. Durant le parcours, des paires de ces ensembles de polygones sont testées. Cependant, des paires de polygones ne sont jamais explicitement testées. Par conséquent, il n'y a aucune information de polygones stockée avec les nœuds des BVHs. Au lieu de cela, la probabilité de l'existence d'une paire de polygones en collision est estimée. Ceci présente deux avantages, d'abord, l'algorithme est vraiment performant en termes de temps d'exécution. L'application peut contrôler le temps d'exécution de l'algorithme en spécifiant la qualité désirée de la détection de collisions. En second lieu, les probabilités peuvent guider l'algorithme aux parties des hiérarchies des BVs qui permettent une évaluation plus rapide de la convergence (figure III.9).

Contrairement aux schémas de parcours traditionnels, l'algorithme est guidé par la probabilité qu'une paire de BVs contienne des polygones en collision [56]:

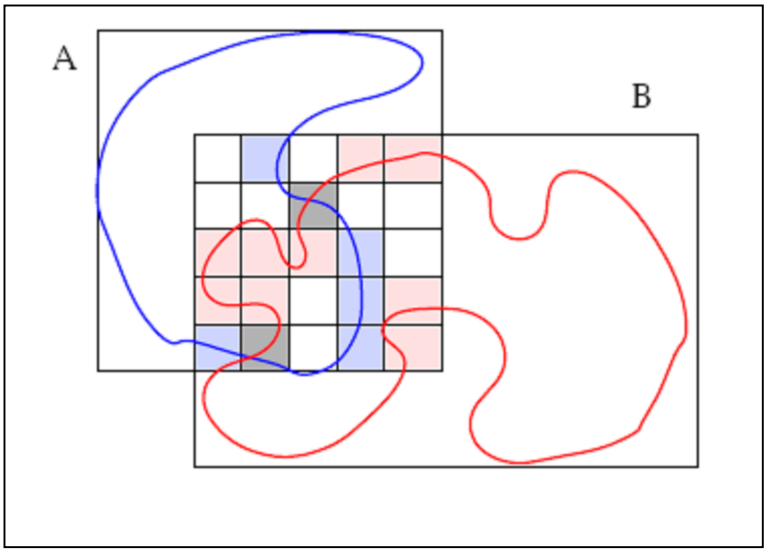


Figure III.9 Détermination du nombre de cellules d'une grille, couvrant A∩B contenant plusieurs polygones appartenant à A et à B [56].

3.5.2 Détection stochastique de collision basée sur les primitives aléatoirement choisies

Une approche naïve de détection stochastique de collisions consiste à choisir des paires aléatoires de caractéristiques en collision comme une initiale supposition de

l'intersection potentielle des régions. Cette méthode peut être encore augmentée en s'assurant que l'échantillonnage couvre les caractéristiques du corps entier et que les caractéristiques sont déjà assez proches. Cependant, ceci n'est pas suffisant pour identifier les régions en collision quand l'objet se déplace ou se déforme. La solution est de considérer la cohérence temporelle. Si une paire de caractéristiques est assez proche à un pas de temps, il peut encore être plus proche dans le prochain. Ceci permet de suivre ces régions en collision durant les pas de temps suivants pendant que les objets sont animés. De plus, ces paires sont considérées pour converger vers les minima locaux de la distance pour identifier efficacement des collisions.

En outre, la cohérence spatiale est également appliquée en maintenant ce minimum local au-dessus des caractéristiques du voisinage. Chaque paire est localement mise à jour à chaque pas de temps, afin de dépister les variations de la distance locale quand les objets se déplacent ou se déforment (figure III.10). Ces paires s'appellent des paires actives. Quand deux paires actives initialement éloignées convergent au même minimum local, une d'entre elles est supprimée.

Une paire est également supprimée si la distance associée est plus grande qu'un seuil indiqué. Ce processus dépiste les régions d'intérêt existantes mais ne détecte pas les nouvelles. C'est un problème sérieux pour les objets déformables non convexes où même le moindre mouvement peut de manière significative changer l'endroit de distance la plus proche [56].

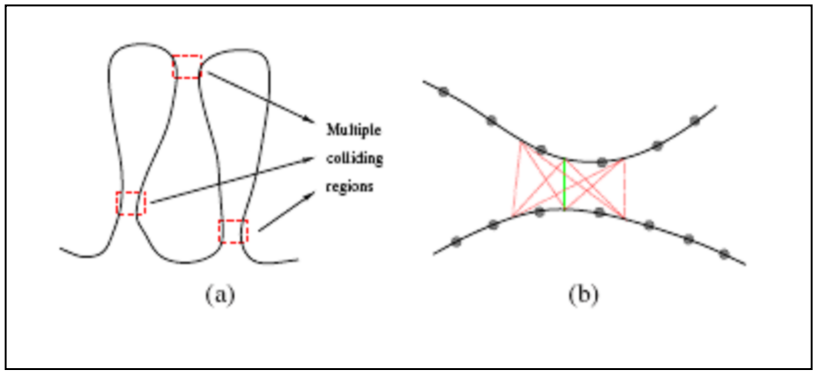


Figure III.10 Détection des collisions ou des auto-collisions entre différents plis des objets minces avec dépistage des minimums locaux de la distance [56].

4. Discussion

Après avoir donné un aperçu sur les techniques de détection de collisions les plus utilisées, nous allons essayer d'évaluer chacune de ces techniques en comparant les

différents avantages et inconvénients de chaque technique. Le choix de la technique de détection de collisions parmi toute cette variété de méthodes dépend surtout des besoins de l'application.

Pour les hiérarchies des volumes englobants, le point le plus important et qui influence beaucoup l'efficacité du processus de détection de collisions, est le choix des volumes englobants utilisés. Cette technique utilisée dans la simulation de vêtements nécessite beaucoup de mises à jour de la hiérarchie, pour la simulation de vêtements.

Les DOPs et spécialement les AABBs qui sont des 6-DOPs, sont les volumes englobants les plus appropriés pour la simulation de vêtements que les autres volumes tels que les OBBs par exemple, ceci est dû au fait que leur mise à jour est plus facile à gérer. Sans oublier que l'utilisation des arbres 4-ary et 8-ary est plus performante que les arbres binaires. Malgré que les objets déformables nécessitent beaucoup de mises à jour, mais la hiérarchie des volumes englobants reste une technique très appropriée pour la simulation des vêtements, si les mises à jour sont faites de manière efficace, et surtout que cette technique peut être utilisée pour la détection des auto-collisions si on lui associe des heuristiques pour accélérer le processus. Le principal avantage des HBVs réside dans leur indépendance par rapport à toute la grille ou la résolution d'image.

Les champs de distances quant à eux, permettent une détection de collisions en temps réel, et permettent même la détection d'auto collisions, mais dans des applications non interactives. Le problème des champs de distance est le temps nécessaire pour leur génération et leur mise à jour et spécialement pour les objets déformables. C'est, d'ailleurs, pour cette raison que les champs de distance ne sont pas appropriés pour les auto-collisions. Ils ne sont performants que pour la détection de collisions entre une pièce de tissu et un objet rigide si le champ de distance peut être bien calculé. Les champs de distance procurent la profondeur de la pénétration et les normales, ce qui est utile pour le traitement de collisions. Cette technique nécessite beaucoup d'espace mémoire et de temps d'exécution, ainsi pour réduire cette nécessité il est possible de réduire la résolution du champ de distance, ce qui produit par la suite une faible précision, mais un certain balancement entre performance et précision est possible.

Les champs de distances n'ont aucun problème avec les changements de topologie des objets déformables.

Pour les techniques de subdivision spatiale, l'avantage majeur est leur simplicité. Le choix de la structure de donnée qui représente et subdivise l'espace 3D influence l'efficacité du

prétraitement, la détection de collisions ainsi que les besoins en mémoire qui s'affecteront aussi par le niveau de subdivision.

Cette technique, utilisée pour la détection de collisions et d'auto-collisions, est indépendante des changements de topologie, ce qui est bien approprié pour les objets déformables ainsi que les objets en mouvement.

Quant aux techniques de l'espace d'image, elles ne nécessitent pas beaucoup de temps en prétraitement ce qui les rendent bien appropriées pour les objets déformables. Cette technique peut être utilisée pour la détection de collisions et d'auto-collisions, et les changements de topologie ne posent pas de problème.

Cette technique procède par une rastérisation ou pixellisation pour détecter les collisions. Par conséquent, elle ne peut pas produire des informations de collisions exactes si elle n'est pas associée à une des autres méthodes. De plus, les informations nécessaires pour le traitement de collisions sont limitées. Comme la précision de cette technique dépend de la résolution de la discrétisation durant le processus de rendu, la performance et la précision peuvent être équilibrées par l'utilisation du matériel dans le rendu, ce qui peut accélérer le processus de détection de collisions.

Les techniques stochastiques quant à elles, reposent sur deux méthodes, la première (average-case), consiste à utiliser les méthodes de probabilité pour estimer la possibilité d'une collision. Pour cette méthode la qualité de la collision peut être spécifiée par l'utilisateur ce qui assure plus de contrôle. La deuxième méthode (méthode basée sur la sélection aléatoire de primitives) devine les paires en collisions par un échantillonnage stochastique. Avec cette méthode la possibilité de collisions entre les régions est réduite. Les deux méthodes sont appropriées pour les objets élastiques volumiques et pour la détection des auto-collisions des objets hautement déformables et fins comme le tissu. Cependant, la première méthode nécessite d'être associée avec d'autres techniques de détection de collisions comme la hiérarchie des volumes englobants. Tandis que la deuxième est totalement indépendante de n'importe quelle autre méthode [60][56].

Le tableau suivant résume les avantages et les inconvénients des différentes techniques de détection de collisions.

Techniques	Avantages	Inconvénients
Technique de la Hiérarchie des volumes englobants.	- La mise à jour des DOPs et des AABBs est facile. - Gestion des auto-collisions possible, associée aux heuristiques. - Appropriée pour le traitement de collisions. - Appropriée pour la détection de collisions continues.	- La mise à jour des OBB difficile à gérer. - Dépendante des changements de topologie. - Pas de compromis entre vitesse et qualité.
Technique des champs de distance.	- Détection de collisions en temps réel. - Appropriée pour le traitement de collisions. - Compromis possible entre vitesse et qualité. - Indépendante des changements de topologie.	- Gestion des auto-collisions non possible. - Non appropriée à la détection de collisions continues.
Technique de subdivision spatiale.	- Simplicité. - Gestion des auto-collisions possible. - Indépendante des changements de topologie.	- Pas de compromis entre vitesse et qualité. - Non approprié pour la détection de collisions continues.
Technique de l'espace d'image	- Gestion des auto-collisions possible. - Indépendante des changements de topologies.	- Non appropriée pour le traitement de collisions. - Pas de compromis entre vitesse et qualité.
Technique des heuristiques	- Gestion des auto-collisions possible. - Indépendante des changements de topologies. - Contrôle de la qualité par l'utilisateur.	- Dépendance avec d'autres techniques de détection de collisions.

5. Le traitement de collision

Dès qu'un morceau de tissu simulé est assez proche d'un objet ou à d'autres parties de lui-même, une interaction a lieu, cette interaction doit être traitée afin de maintenir une

simulation réaliste. Cette réponse aux collisions détectées est utilisée pour éviter les interpénétrations des objets et pour modéliser le frottement cinétique et statique.

La particularité des traitements de collisions pour les objets déformables réside dans le nombre important de collisions qui surviennent dans un pas de temps, la deuxième particularité des objets déformables qui rend le traitement crucial est l'épaisseur très petite du tissu où même les interpénétrations banales ou marginales peuvent causer des artefacts qui dérangent la visibilité et entrainent des difficultés lors du traitement de collisions.

Le traitement de collisions est principalement utilisé pour imposer une distance minimale entre les deux éléments de surface en contact [55].

Pour les scènes animées, une méthode efficace de traitement de collisions est cruciale. En particulier, pour la stabilité numérique et la performance dans le contexte où de grands pas de temps doivent être préservés et aucune raideur supplémentaire ne devrait être introduite dans le système.

Pour l'étape de traitement de collisions dans le processus de la simulation, plusieurs méthodes ont été proposées. Ces méthodes peuvent être classées, en deux classes: approches physiques et approches géométriques. Les approches de traitement de collisions qui modélisent le champ de la force répulsive entre les objets sont dites méthodes physiques. Tandis que les approches qui limitent les degrés de liberté du mouvement du corps, ou qui modifient directement les positions, les vitesses, et les accélérations sont dites approches géométriques.

5.1 Approches physiques de traitement de collisions

L'idée derrière le traitement de collisions physiques est de modéliser le champ de la force potentielle répulsive F entre les objets solides de la distance r avec $F \sim r^{12}$, qui évite en réalité l'intersection. Comme les forces peuvent être directement intégrées dans le modèle mécanique et les équations différentielles résultantes peuvent être résolues en conséquence, les approches physiques sont les plus formelles pour traiter les collisions. L'avantage de cette classe de méthodes basées sur les champs de force réside dans le fait qu'elle est très proche des lois physiques. Cependant, il a été démontré qu'elles souffrent de plusieurs difficultés, la première est de modéliser le contact entre les objets en collision, à savoir assurer une distance minimale de collision. Modéliser directement des champs de force de portée très courte est impossible en utilisant les grands pas de temps de 10^{-3}s à 10^{-1}s, habituellement utilisés dans les solveurs numériques pour les simulations du tissu. Par conséquent, il est nécessaire de modéliser le champ de force avec les champs de

potentiel d'une portée plus longue. Ceci, cependant, peut causer des résultats non réalistes de la simulation, du fait que la distance minimale assurée entre les objets en collision est trop grande. En outre, les champs de force, induisent toujours une raideur dans les équations différentielles qui réduit les performances et la stabilité des intégrateurs numériques. Dû à ces inconvénients, le traitement de collision physique est utilisé pour la simulation de vêtements quand les pas de temps utilisés sont réduits.

Dans [79], Moore et Wilhelms ont proposé l'utilisation de ressorts répulsifs pour la séparation des objets rigides simulés quand ils deviennent trop proches. De leur part, dans [80], Terzopoulos et Fleischer ont gardé une formulation continue en utilisant un champ de force de déviation autour des objets déformables en collision. D'autre part, Lafleur et al dans [81] ont proposé un champ de force autour de chaque triangle en collision dont la force du champ de force dépend de la vitesse, des normales et de la distance entre les objets en collision. Une approche hybride combinant un champ de force contrôlé par une méthode de traitement de collisions géométriques est proposée par Bridson et al. [76]. Ils limitèrent les capacités des forces de collision pour éviter d'introduire une raideur dans l'équation différentielle. Afin de résoudre les situations compliquées de l'auto-collision, Baraff et al. [51], utilisèrent des forces entre les particules. Les parties du tissu qui ont intersecté d'autres parties et qui se trouvent du mauvais côté sont séparés par des forces attractives, tandis que les particules proches qui sont encore sur le bon côté se repoussent mutuellement [60].

5.2 Approches géométriques de traitement de collision

Le traitement de collisions géométriques tente de simuler l'effet des principes de la physique, sans l'utilisation explicite des forces de répulsion. Les algorithmes proposés modifient directement l'état géométrique, c.-à-d. les positions, les vitesses et l'accélération du vêtement simulée pour contraindre le mouvement de l'objet dans certaines directions.

Le principal avantage de cette approche est la séparation de la réponse de collision et de l'intégration numérique. Par conséquent, la performance des solveurs des équations différentielles n'est pas réduite en raison de la rigidité induite. Mais cette séparation peut être vue comme un désavantage parce que rien ne garantit la compatibilité entre le changement de l'état géométrique et le bon résultat d'un champ de forces répulsives. Aussi, la conservation de l'énergie et le moment devraient être respectés lors de la manipulation manuelle de l'état du système. Eberhardt et al. ont proposé dans [82] une méthode pour

modifier les positions et les vitesses, cette méthode est basée sur un test d'intersection des rayons. Si le rayon de la position initiale à la nouvelle position après un pas de temps, intersecte un obstacle, alors la normale de la surface au point d'intersection est calculée. En utilisant la normale de la surface, ils déplacent les particules du tissu vers un point qui a une distance minimale au-dessus de l'obstacle. En outre, la vitesse est altérée, ce qui empêche la particule de se déplacer plus loin en direction de l'obstacle. Grâce à cette modification de la vitesse, la friction et l'élasticité du contact sont également modélisées. Platt et Barr [83], utilisèrent des contraintes de réaction pour permettre aux animateurs de contrôler le mouvement et d'éviter l'interpénétration entre les objets simulés. Volino et Magnenat-Thalmann [63, 64, 65] proposèrent un algorithme de modification des positions, des vitesses et des accélérations en fonction de la distance entre les objets en collision.

Bridson et al. [76] utilisèrent une approche hybride combinant les forces répulsives, ainsi que les changements de la vitesse dus aux impulsions ajoutées. Les objets proches les uns des autres à un certain seuil sont séparés en utilisant le champ de forces alors que les objets qui s'interpénètrent au cours d'un pas de temps sont traités en utilisant les impulsions [60].

6. Les auto-collisions

Au cours de la simulation, le vêtement à tendance à entrer en collision avec son environnement, et avec lui-même (auto-collisions). Afin de garantir la réalité de la simulation, il est nécessaire de détecter ces collisions et de donner des réponses au bon moment, autrement, la pénétration se fera et altérera le réalisme de la simulation. Le processus d'auto-collision comprend la détection de collisions et le traitement de collisions. La détection des collisions vise à trouver des collisions et le calcul des paramètres pertinents, tandis que le traitement de collisions vise à déplacer correctement les objets intersectés après la collision en fonction du point de collision et d'autres paramètres pour reproduire l'effet dynamique réel.

Du fait que la simulation de vêtements nécessite le temps réel, la vitesse de la détection des auto-collisions et leur traitement peut directement influencer la vitesse de la simulation. Par conséquent, il est essentiel de de mettre au point un bon algorithme de détection et de traitement de collisions en respectant à la fois le réalisme et le temps réel.

David Baraff [41] a analysé le réseau des collisions du modèle de tissu à chaque étape et a fait les traitements correspondants. Sa méthode permet d'éviter la présence de

certaines structures désordonnées entraînées par erreur, ce qui peut plus tard influencer le processus de la prochaine collision.

Mathieu Desbrun [43] joint une couche de surface implicite sur la structure de base des objets pour simuler quelques objets spécifiques. Il peut générer rapidement les surfaces de contact précises et les déformations locales.

Doug L. James et al [48], ont principalement fait des recherches dans la simulation interactive et le modèle physique avec une haute crédibilité, et ont proposé une méthode de précalcul pour les modèles déformables dérivés par les données.

Robert Bridson et al [76] ont focalisé leurs recherches sur le traitement des collisions dans la simulation de modèles avec des caractéristiques de frottement. Ils ont proposé un algorithme rapide pour le processus de collision des tissus, incluant les collisions avec les objets rigides et l'auto-collision. Leur méthode possède une bonne évolutivité ainsi qu'une bonne robustesse.

Andrew Selle et al [77] ont centré leurs recherches sur la simulation de tissu à haute résolution et ont proposé un schéma de détection de collisions basé sur l'historique [75].

7. Détection de collisions continues

Pour la simulation de deux maillages triangulaires déformables, les tests de collision sont généralement effectués à chaque pas de temps. Mais les tests de proximité ne peuvent pas éviter que deux objets entrent en collision dans un seul pas de temps. Afin de détecter toutes les collisions, des tests de détection de collisions continues doivent être effectués. Ceux-ci nous permettront de prévenir l'interpénétration des objets indépendamment de la taille du pas de temps, de la vitesse des objets, et de l'épaisseur du matériau. Pour chaque paire de primitives, il faut détecter si elles entrent en collision dans le pas de temps courant, et à quel moment cette collision se produit.

Deux problèmes se posent lorsque la détection de collisions continues est utilisée:

- Les collisions qui se produisent plus tard dans le temps peuvent être détectées plus tôt dans le processus de détection de collisions. Pour obtenir une réponse précise aux collisions et pour un comportement plausible des objets simulés, les collisions devraient être traitées dans l'ordre chronologique correct.
- Les réponses aux collisions peuvent causer de nouvelles collisions secondaires.

Ces collisions peuvent engendrer des interpénétrations, si elles ne sont pas détectées et résolues.

Afin de remédier au problème des collisions secondaires causées par les réponses aux collisions précédentes, un processus itératif peut être utilisé. Une deuxième méthode de traitement des collisions dans le bon ordre chronologique peut être appliquée, où pour chaque triangle, seule la première collision survenue devrait être prise en compte. La combinaison de ces deux techniques permet un traitement robuste et plausible de toutes les collisions qui se produisent dans un pas de temps [74].

8. Quelques travaux de simulation de vêtements impliquant la gestion de collisions

Plusieurs travaux durant ces deux dernières décennies ont été élaborés par les pionniers du domaine. Ces travaux adoptent plusieurs techniques qui visent à accélérer la simulation de vêtement ainsi qu'à améliorer sa qualité et sa performance. Cependant, les performances propres au temps réel n'ont pu être obtenues que sur des cas limités, ce qui laisse cet axe de recherche toujours exploitable et l'évolution des travaux se poursuit jusqu'à nos jours.

Dans ce qui suit, nous allons citer quelques travaux élaborés au cours de ces dernières décennies. Parmi ces travaux :

- **Les travaux de Baraff et Witkin**

Dans [19], les auteurs ont introduit un modèle de simulation dynamique des objets déformables sous la contrainte de non-pénétration. Les collisions ne sont pas considérées comme des collisions instantanées. Lorsque deux objets déformables se heurtent, un nombre fini de sommets de l'un de ces objets sont en contact avec l'autre corps et doivent être considérés. Pour simuler une collision entre deux objets déformables, les auteurs suivent l'approche suivante:

1. Quand un point pénètre un corps étranger, il est poussé vers l'extérieur.
2. Sa vitesse par rapport au corps étranger est annulée.
3. Définition des contraintes physiques du corps, y compris la rigidité affectera les voisins du point qui a engendré la pénétration.
4. Enfin, le corps va être comprimé localement, puis reviendra à son état de repos. Ce n'est que lorsque le volume aura retrouvé son état de repos que le point de collision sera séparé du corps.

Ce travail décrit de façon très rigoureuse les modèles déformables et le traitement de collisions entre ces modèles.

Dans [41], les auteurs ont présenté une approche pour simuler les vêtements. Cette méthode peut utiliser et de manière stable de larges pas de temps entre deux étapes d'intégration implicite, en couplant une nouvelle technique qui renforce les contraintes sur les particules du vêtement avec la méthode d'intégration implicite. Pour la phase de détection de collisions, les auteurs se réfèrent à leur travail [19] qui est adéquat pour la détection de collisions entre les objets déformables comme les vêtements le sont. L'auto collision est détectée en vérifiant les intersections des paires (p, t) et (e_1, e_2) où p et t sont respectivement les particules et les triangles du vêtement et e_1 et e_2 sont les arêtes des triangles du vêtement.

Compte tenu d'un état précédent connu du tissu, ils ont choisi d'appliquer un mouvement linéaire pour les particules du tissu à l'état actuel et vérifient la présence d'intersections particule/triangle ou d'intersections arrête/arrête. Pour éviter un nombre de comparaisons de l'ordre de $O(n^2)$, ils utilisèrent une technique cohérente des volumes englobants. Lorsque des collisions entre un sommet du tissu et un triangle, ou deux arrêtes, sont détectées, ils introduisirent une force de ressort amortie pour repousser le tissu. Une force dissipative tangente au contact est également appliquée, luttant contre tout mouvement de glissement. Leur système détecte les collisions entre les particules de tissu et des objets solides par des tests individuels de chaque particule du tissu par rapport aux faces de chaque objet solide. Les faces de l'objet solide sont regroupées dans une structure hiérarchique de volumes englobants (arbre), dont les feuilles de l'arbre étant les faces du solide. L'arbre est créé par un simple partitionnement récursif le long des axes de coordonnées.

Dans [51], Baraff et Witkin se sont intéressés à la collision vêtement/vêtement ou à l'auto collision, en se basant sur un algorithme d'analyse d'intersection globale du maillage du vêtement dans chaque pas de la simulation. Le traitement de ces collisions est simple. Une fois les intersections entre les maillages du tissu ont été trouvées en utilisant l'algorithme GIA (analyse d'intersection globale), les auto-collisions du tissu étant traitées en introduisant les forces d'interaction entre une particule du tissu p et le plus proche triangle T du tissu.

- **Les travaux de Thalmann et son équipe**

Dans [52] Thalmann, Volino et Courchesne ont eu pour objectif principal dans leur travail d'être capables de simuler tout type de surface sans imposer de restrictions sur la forme ou la géométrie de l'environnement. Le travail décrit les différents paramètres par rapport aux différents types de collisions qu'il traite qui sont :

- Proximités: triangle/sommet, arête/arête, arête/sommet ou sommet/sommet.
- Interférences: se produit chaque fois qu'une arête traverse un triangle.

Ils ont aussi utilisé plusieurs approches pour détecter la collision :

- Proximités restantes : garder une trace des proximités entre les triangles de sorte que lorsqu'elles se produisent encore, ils pourront être détectés.
- Suivi cinématique: quand une proximité est détectée, ils considèrent les vitesses courantes des triangles et en déduisent si une collision est survenue précédemment.
- Vérification et correction de la cohérence: les approches précédentes ne sont pas déterministes, donc l'ajout d'une vérification de la cohérence est nécessaire. Elle est basée surtout sur les régions de collisions.

Dans [53], Thalmann et al ont présenté un système plus général capable de créer des vêtements autonomes qui peuvent être portés par n'importe quel humain virtuel. Le système entier est divisé en plusieurs parties dont :

- La structure de données qui contient toutes les informations du système.
- Le moteur de détection de collisions qui calcule les collisions entre tous les objets de la structure.
- Le moteur de simulation mécanique qui anime les objets en fonction de différents modèles mécaniques pour les objets rigides ou déformables.

Le système tire profit d'un algorithme efficace de détection de collisions qui est basé sur la construction d'hiérarchies géométriques de surfaces de régions et prend l'avantage de la courbure de la surface à l'intérieur et entre les régions adjacentes de surfaces pour optimiser la détection d'auto-collisions. La construction de la hiérarchie est effectuée au début de la simulation. Après cela, seules les boîtes englobantes et les normales sont recalculées pour tous les éléments de la hiérarchie. Et la détection est effectuée au sein d'un élément ou entre deux éléments adjacents que si leur courbure est compatible avec l'existence de collisions.

Cet algorithme permet au programme de faire un large usage de la détection d'auto-collisions sans faire aucune hypothèse sur les parties qui peuvent entrer en collision, offrant ainsi une grande flexibilité sur les situations difficiles impliquant le plissement complexe et le froissement. Associé au moteur de simulation mécanique, cet algorithme de détection de collisions conserve son efficacité et sa robustesse dans la plupart des types de situations rencontrées dans la simulation des vêtements.

Dans [54] Thalmann a présenté les différentes progressions et évolutions effectuées dans leurs recherches sur la simulation des vêtements virtuels, les cheveux et la peau pour des humains synthétiques. Dans leur présentation, ils ont invoqué les différentes techniques de gestion de collisions utilisées jusqu'à 1998. Au lieu de considérer les collisions comme étant dynamiques, elle a proposé de repousser les objets considérés en utilisant des forces élevées et discontinues, ce qui ne nécessite des pas de temps très petits pour la précision de calcul, les collisions sont plutôt prises en compte dans une étape distincte comme une résolution des contraintes géométriques et cinématiques indépendantes du calcul mécanique résultant des forces continues, telles que l'élasticité, et de la gravité et le vent. Dès que deux éléments entrent en collision, le transfert de quantité de mouvement est effectué en fonction des lois de la conservation mécanique, et en tenant compte des effets de friction et de rebond. De cette manière, toutes les collisions sont traitées indépendamment, pendant un pas de temps unique et commun. Lorsque les éléments sont impliqués dans plusieurs collisions, comme pour les vêtements multicouches, le traitement de collision est effectué itérativement jusqu'à ce que toutes les contraintes de collision soient résolues. Cette technique permet également la propagation de l'effet de collision par les différentes couches du vêtement.

En fonction de la façon dont l'objet (vêtement) est représenté, différentes techniques ont été développées pour résoudre efficacement le problème de détection de collisions en utilisant des méthodes basées sur la subdivision de l'espace telle que la voxelisation ou l'octree ou la technique de la hiérarchisation ou encore le suivi de la plus petite distance ou des techniques mathématiques adaptées aux surfaces paramétriques courbées.

Dans le travail de l'équipe de Thalmann [55], les auteurs ont proposé une approche unifiée pour la déformation du tissu et du corps. Ce système est capable de gérer à la fois la peau et la déformation de tissus avec le traitement de collisions. Ces déformations sont gérées par un système de particules. Dans de telles situations où les collisions doivent être détectées entre les maillages polygonaux animés, les techniques des volumes englobants basées sur la hiérarchie des polygones construits sur les maillages semblent les meilleures méthodes, comme la structure hiérarchique peut être maintenue constante pendant que les surfaces sont en mouvement, et seuls les volumes englobants doivent être recalculés entre chaque pas d'animation. Le traitement de collisions est principalement utilisé pour imposer une distance minimale entre les deux éléments de surface en contact. Le traitement des contraintes de collisions en utilisant la correction cinématique sur les éléments contraints. Plutôt que de calculer les forces de collisions à travers la cinématique inverse de la loi de

conservation dynamique, ils ont directement intégré les contraintes par des corrections de positions sur les sommets concernés, selon la conservation de la force d'impulsion (moment). Les positions sont corrigées selon les lois de conservation mécanique pour répondre aux contraintes de précision. Une correction immédiate de la position des sommets concernés, prise en compte avant le processus de simulation dynamique, vise à refléter les effets immédiats de la contrainte. Une force correctrice a pour but d'atténuer ou annuler la différence de forces entre les sommets contraints, afin de maintenir les contraintes cinématiques imposées. Cette correction est faite lors de l'intégration. Le réglage de la position se fait sur de nouvelles positions après l'intégration.

La figure III.11 donne un aperçu sur la méthode de traitement de collision.

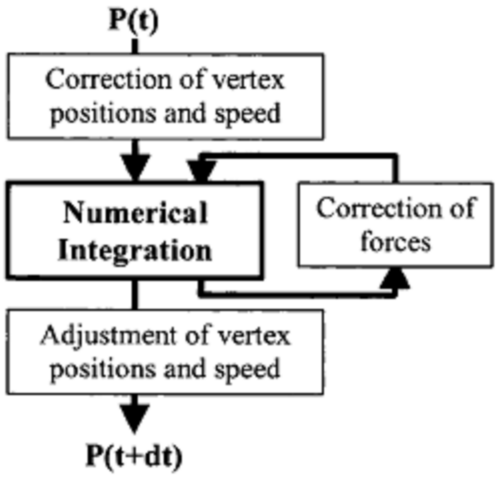


Figure III.11 Le traitement de collision [55]

Dans un autre travail de l'équipe de Thalmann [56], les auteurs ont réuni les résultats d'autres chercheurs comme Heidelberger, Zachmann et Teschner, qui avait pour but de résumer les recherches dans le domaine de la détection de collisions pour des objets déformables. Diverses approches basées sur des hiérarchies de volumes englobants, champs de distance, et la subdivision spatiale, les techniques de l'espace d'image et des méthodes stochastiques sont discutées. Des applications de ces techniques sur la simulation chirurgicale et la simulation des tissus sont présentées.

Le travail présenté dans [59] présente une approche basée données pour le processus temps réel de la simulation de vêtements. Le travail commence, avant la simulation en temps réel, en analysant le comportement du tissu par rapport au mouvement du squelette sous-jacent à partir d'une séquence de présimulations du tissu obtenu en utilisant n'importe

quel simulateur hors connexion de haute qualité. Une méthode de détection de collisions basée données est appliquée, en utilisant la séquence présimulée pour localiser les tests de collisions dans les régions voisines, qui ont une forte probabilité d'entrer en collision.

- **Autres travaux**

Dans[57]n T.Vassilev et al présentèrent une technique rapide pour animer des vêtements sur humains virtuels en mouvement. Ils ont exploité un modèle masse-ressort pour la simulation du tissu, en appliquant une nouvelle approche de modification de direction de la vitesse pour surmonter sa surélasticité. L'algorithme de détection et la réponse de collision tissu-corps repose sur des tests d'interférence sur l'espace image. Un matériel graphique moderne est utilisé non seulement pour calculer les profondeurs du corps, mais aussi pour interpoler les vecteurs et les vitesses et les normales de chaque sommet.

Baciu et Wang [61], ont développé une méthode de détection de collisions qui réalise un taux d'interactivité pendant la détection des auto-collisions des objets hautement déformables. La méthode utilise une nouvelle technique pour la génération dynamique d'une hiérarchie de volumes englobants pour les tissus.

Dans le travail de Mezger et al [63], l'efficacité de la hiérarchie des volumes englobants est améliorée par des techniques adaptées pour la construction et le parcours de ces hiérarchies. Un ensemble d'heuristiques étendu est décrit qui permet de détailler la hiérarchie. L'inflation orientée des volumes englobants permet de détecter les proximités avec un minimum de coût. Finalement, la distance entre les facettes du maillage est calculée avec précision ainsi que les contraintes de réponse aux collisions.

Teschner, Heidelberger et al, dans [66], ont proposé une nouvelle approche de détection de collisions et d'auto-collisions dynamiques des objets déformables qui se composent de tétraèdres. Cet algorithme utilise une fonction de hachage pour la compression potentielle d'une grille spatiale régulière infinie. Bien que la fonction de hachage ne fournit pas toujours une organisation unique de cellules de la grille, elle peut être générée de manière très efficace et ne nécessite pas de structures de données complexes, telles que les octrees ou les BSP. L'algorithme permet de détecter les collisions et les auto-collisions dans des environnements allant jusqu'à 20000 tétraèdres en temps réel. Bien que l'algorithme fonctionne avec des maillages tétraédriques, il peut être facilement adapté à d'autres primitives d'objets, telles que des triangles.

Bridson et Fedkiw, dans [50], présentèrent plusieurs méthodes essentielles à l'appariement du comportement des vêtements simulés et les comparer avec les vêtements réels. En utilisant un mélange de schémas d'intégration explicite/implicite, un modèle de flexion physiquement correcte pour la mise en forme des plis, une technique de prévision pour favoriser le développement de détail des régions en contact, une méthode de post-traitement pour le traitement des collisions du tissu qui préserve les plis, et un mécanisme de contrainte dynamique qui aide à contrôler le pliage à grande échelle. L'objectif commun de toutes ces techniques est de produire une simulation de tissu avec de nombreux plis et rides pour améliorer le réalisme.

Choi et al [68] a présenté un schéma de détection et résolution de collisions rapide et robuste pour les objets déformables, en utilisant un nouveau modèle amélioré de la hiérarchie des surfaces sphériques implicites. Le taux de pénétration, la profondeur et la séparation des critères de distance peuvent être ajustés en fonction de la demande selon la tolérance d'erreur spécifique à l'application. Leurs expériences comparatives montrent que leur méthode est sensiblement plus rapide que les algorithmes existants pour la simulation des objets déformables. Cette approche a permis de réaliser une simulation en temps réel.

Manocha et al [69] présentèrent une nouvelle approche pour accélérer la détection de collisions pour les modèles déformables. La formulation s'applique à tous les modèles triangulés et réduit considérablement le nombre de tests élémentaires entre les éléments du maillage, c'est-à-dire, sommets, arêtes et faces. Ils introduisent la notion de représentatif triangle, qui est un type géométrique de triangles augmentés avec des informations sur les fonctionnalités du maillage, afin d'obtenir de meilleures performances pour le processus de détection de collision. L'approche qui en résulte peut être combinée avec des hiérarchies des volumes englobants pour la gestion des auto-collisions. Jund, Cazieret et Dufourd [70] ont proposé une méthode basée sur une subdivision volumique de l'environnement, couplée à un mécanisme de prédiction. L'environnement est subdivisé en un ensemble de cellules convexes enrichies par une structure topologique forte permettant des requêtes de voisinages optimales. Ce système de prédiction utilise la cohérence temporelle et les propriétés des cellules pour optimiser le nombre de tests d'intersection à effectuer à chaque pas de temps.

Manocha et al, dans [71], présentèrent un algorithme de collision qui utilise des filtres de non-pénétration pour améliorer la performance de la détection des collisions continues (CCD). L'idée est d'utiliser un simple filtre efficace qui réduit le nombre de faux positifs et les tests élémentaires entre les primitives. Ce filtre est dérivé des conditions de

coplanarité et peut être facilement combiné avec d'autres méthodes utilisées pour accélérer la détection de collisions continues (CCD).

Tang, Manocha et Tong [72], ont proposé un nouvel algorithme parallèle pour la détection rapide de collisions continues (CCD) entre les modèles déformables en utilisant les processeurs multi-core. Ils utilisèrent une représentation hiérarchique pour accélérer la détection et en y proposant un algorithme incrémental qui exploite la cohérence temporelle entre les trames successives. Leur formulation distribue le calcul entre plusieurs cores grâce à une décomposition fine de positions granulées. Ils présentèrent également des techniques efficaces pour réduire le nombre de tests élémentaires et analyser l'extensibilité de l'approche.

Govindaraju, et al [73] ont présenté un nouvel algorithme pour détecter avec précision tous les contacts, y compris l'auto-collision, entre les modèles déformables. Ils ont calculé la décomposition chromatique d'un maillage en primitives non adjacentes en utilisant des algorithmes de coloration de graphes. La décomposition chromatique permet de vérifier les collisions entre primitives non adjacentes en utilisant un algorithme de Culling en temps linéaire. Cet algorithme permet de réduire significativement le nombre de faux positifs, il est utilisé pour vérifier les collisions entre les modèles déformables complexes composés de dizaines de milliers de triangles pour la modélisation de tissu et de simulation médicale.

III. La multi résolution (les niveaux de détail)

Réaliser un niveau de détail (NDD) d'un élément consiste à en construire une version moins complexe géométriquement, c'est-à-dire comportant moins de facettes. Il s'agit donc d'afficher des versions plus au moins dégradées de l'objet selon certains critères tel que la distance entre la caméra et l'objet ou la taille de l'objet à l'écran.

La complexité des environnements virtuels influence le temps de traitement informatique et l'affichage. Plus on ajoute de complexité visuelle à une scène, plus long sera le temps du rendu [46]

idéalement, les algorithmes de créations des niveaux de détails doivent posséder plusieurs qualités :

- Pouvoir créer une suite de représentations de plus en plus simplifiées ;
- Pouvoir mesurer et contrôler de manière intuitive le degré de simplification ;

- Tenir compte des parties significatives des objets pour les préserver dans les représentations simplifiées: intuitivement, il s'agit de trouver les parties de l'objet que l'observateur percevra le plus facilement ;
- Pouvoir faire varier le degré de simplification à travers l'objet pour, par exemple, simplifier davantage les parties peu importantes ou lointaines d'un objet ;
- Les différentes résolutions que l'on veut mélanger doivent avoir, dans une certaine mesure, le même comportement dynamique ;
- Les différentes résolutions doivent pouvoir cohabiter, c'est-à-dire que les transitions entre les niveaux de détails ne doivent pas être visibles [90].

Par conséquent, les principaux problèmes liés à la multi résolution sont la création des LODs et leur gestion.

1. Création des LODs

Il existe deux grandes catégories pour la création des LODs, les algorithmes basés sur la géométrie et ceux basés sur la scène.

1.2 Simplification orientée géométrie

Pour la simplification orientée géométrie, nous pouvons en citer trois qui sont catégorisés selon plusieurs critères dont :

- La simplification selon les sommets : tous les points à l'intérieur d'un volume sont regroupés ;
- La simplification selon les arêtes : tous les bords d'une longueur inférieure à une longueur donnée sont éliminés ;
- Simplification basée sur les angles : les bords contiennent un angle inférieur à une valeur donnée sont éliminés ;
- Simplification basée sur la surface d'une facette : les facettes dont la surface est inférieure à une valeur donnée sont éliminées ;
- Simplification basée sur la normale à une facette : les facettes dont les normales sont presque parallèles sont éliminées.

1.2.1 Subdivision adaptative

Cette méthode commence avec un modèle de base très simple et le subdivise récursivement en ajoutant des détails à certains endroits du modèle à chaque étape.

L'algorithme s'arrête lorsque le modèle approxime le modèle original selon un degré spécifié par l'utilisateur.

1.2.2 Réduction géométrique

Cette méthode part d'un modèle d'origine en simplifiant les faces ou les sommets récursivement. L'algorithme s'arrête lorsqu'il ne peut plus retirer de géométrie et lorsque le modèle satisfait un degré d'approximation spécifié par l'utilisateur.

1.2.3 Échantillonnage

Cette méthode fonctionne de manière opposée à la méthode précédente au sens où elle cherche à trouver les primitives à conserver plutôt que celles à retirer. Elle effectue un échantillonnage de la géométrie du modèle original, soit en choisissant arbitrairement un certain nombre de sommets, soit en englobant le modèle dans une grille tridimensionnelle, et en échantillonnant chaque boite de la grille. L'algorithme essaye alors de créer un modèle simplifié qui soit proche des données échantillonnées.

1.3 Simplification orientée scène

Cet algorithme cherche à simplifier des régions de la scène plutôt que les objets eux-mêmes. Ainsi, dans une phase de précalcul, ils décomposent récursivement la scène en zones 3D, pour obtenir une description hiérarchique, chaque niveau étant plus détaillé que le précédent. Ensuite, à chaque nœud de la hiérarchie, une représentation simplifiée de la sous-hiérarchie est produite. Cette représentation sera utilisée si les critères de sélection sont satisfaits [32].

2. Gestion des LODs

Si deux images successives utilisent deux représentations différentes d'un objet, il est possible d'échanger immédiatement les représentations entre les deux images. Toutefois, cela peut provoquer des sauts perceptibles par l'utilisateur, si ces deux représentations sont utilisées alternativement dans plusieurs images successives.

Pour pallier ce problème, certains systèmes remplacent progressivement les représentations par transparence en faisant varier l'opacité des objets. Mais cela est coûteux puisque deux représentations de l'objet sont présentes en mémoire. La solution consiste à utiliser, au sein de l'algorithme de sélection, un critère d'hystérésis qui produit

une inertie dans les changements des LODs, en interdisant tout changement dans les *n* images suivant une transition [90].

IV. Conclusion

Dans la simulation dynamique, la détection de collisions est essentielle pour un comportement réaliste des objets simulés. Des solutions efficaces pour la détection de collisions des corps rigides ont été développées, mais dans la simulation d'objets déformables, la détection de collisions est encore considérée comme le goulot d'étranglement, surtout pour les objets hautement déformables comme le tissu. Plusieurs travaux qui s'inscrivent dans le domaine de la simulation de vêtements traitent le problème de la gestion de collisions, en utilisant différentes techniques et approches, qui visent toutes à garantir le compromis entre la performance de la simulation et la précision de la détection. Nous avons commencé ce chapitre par donner une idée générale sur la simulation des vêtements, ses méthodes et les différentes techniques appliquées. Ensuite, nous avons abordé la partie concernant la gestion de collisions dans la simulation de vêtements, où nous avons expliqué les méthodes de détection de collisions les plus utilisées, puis nous avons essayé d'analyser ces méthodes en indiquant leurs avantages et inconvénients, pour faciliter le choix d'utilisation de ces différentes méthodes. Nous avons ensuite donné un aperçu sur les auto-collisions et les collisions continues, suivies des différentes approches de traitement de collisions. Et pour finir cette deuxième partie du chapitre, nous avons abordé quelques travaux de simulation de vêtements impliquant la gestion des collisions, durant ces deux dernières décennies. Enfin, nous avons donné un aperçu sur la notion de multi résolution.

Chapitre IV - Intégration de la multi-résolution dans la détection de collisions basée sur la hiérarchie des volumes englobants :

Application à la simulation de vêtements

1. Introduction

La détection de collisions est l'étape la plus coûteuse en termes de temps de calcul dans un processus de simulation, en particulier pour les objets déformables. Pour pallier ce problème, le processus de détection de collisions peut être décomposé en deux grandes étapes: La première consiste en la détection de collisions grossières qui ne permet pas de déterminer exactement les entités en collision, mais qui donne une idée grossière de la région de l'objet sur laquelle une collision peut survenir. En d'autres termes, elle détecte les collisions potentielles. La deuxième c'est la détection exacte, celle-ci permet de connaitre exactement, si des entités en collision existent.

La première phase se caractérise par des méthodes accélératrices qui permettent de réduire le coût de la détection de collisions, en économisant sur le nombre de tests établis, ce qui conduit à réduire la taille de la mémoire occupée ainsi que le temps d'exécution. Parmi ces méthodes, celles basées sur la hiérarchie des volumes englobants.

L'idée des hiérarchies des volumes englobants (BVHS) consiste à partitionner chaque objet en un ensemble de primitives englobées par des volumes. Les BVHS accélèrent la détection de collisions de deux façons: d'abord, par une détection de collisions entre volumes englobants (BVs), qui est beaucoup plus rapide que la vérification de la collision entre les objets proprement dits. De plus, la hiérarchie est utilisée efficacement pour éviter des tests de collisions inutiles dans des zones où aucune collision ne peut se produire [60]. La détection peut être mieux accélérée en utilisant plusieurs représentations des objets selon le niveau de

détails voulu. Ceci peut être obtenu en combinant la technique des BVHs avec celle de la multi résolution.

La multi résolution consiste à disposer d'une résolution variable au cours de l'animation. De plus, avoir à disposition, à tout instant, différentes représentations du même mouvement et choisir en permanence celle qui convient le mieux. [90]

La deuxième phase permet de calculer de façon exacte les entités en collision une fois que la phase de détection grossière ait donné une approximation de la zone de contact. Dans ce cas, des algorithmes de calculs de distances et d'intersections seront utilisés pour la détection de collisions. Une liste de paires de primitives en intersection sera communiquée au processus de traitement de collisions.

Dans ce chapitre, nous allons exposer notre travail, qui consiste à gérer les collisions en utilisant la méthode d'accélération de la détection basée sur la hiérarchie des volumes englobants.

Nous commençons par présenter la méthode de la hiérarchie des volumes englobants que nous avons choisie, en présentant les détails de notre approche qui consiste à créer la hiérarchie des sphères englobantes ainsi que les différentes fonctions qui lui sont associées. Ensuite, nous expliquons comment nous allons introduire la multi résolution avec la hiérarchie des sphères englobantes pour accélérer la détection des collisions. Nous finissons par décrire la méthode de traitement de collisions choisie.

2. Présentation de l'approche

La simulation de tissu est une tâche exigeante qui requiert la résolution de divers problèmes dans les domaines de l'informatique, des mathématiques et de la physique. Créer un modèle pour la simulation de vêtements en temps réel nécessite une action sur les aspects principaux qui sont la modélisation des vêtements, leurs comportements et la gestion des collisions. Dans ce qui suit, nous donnons un aperçu sur les étapes entremêlées du pipeline de simulation. Notre travail, au sein de ce système, se focalise sur le module de la gestion des collisions.

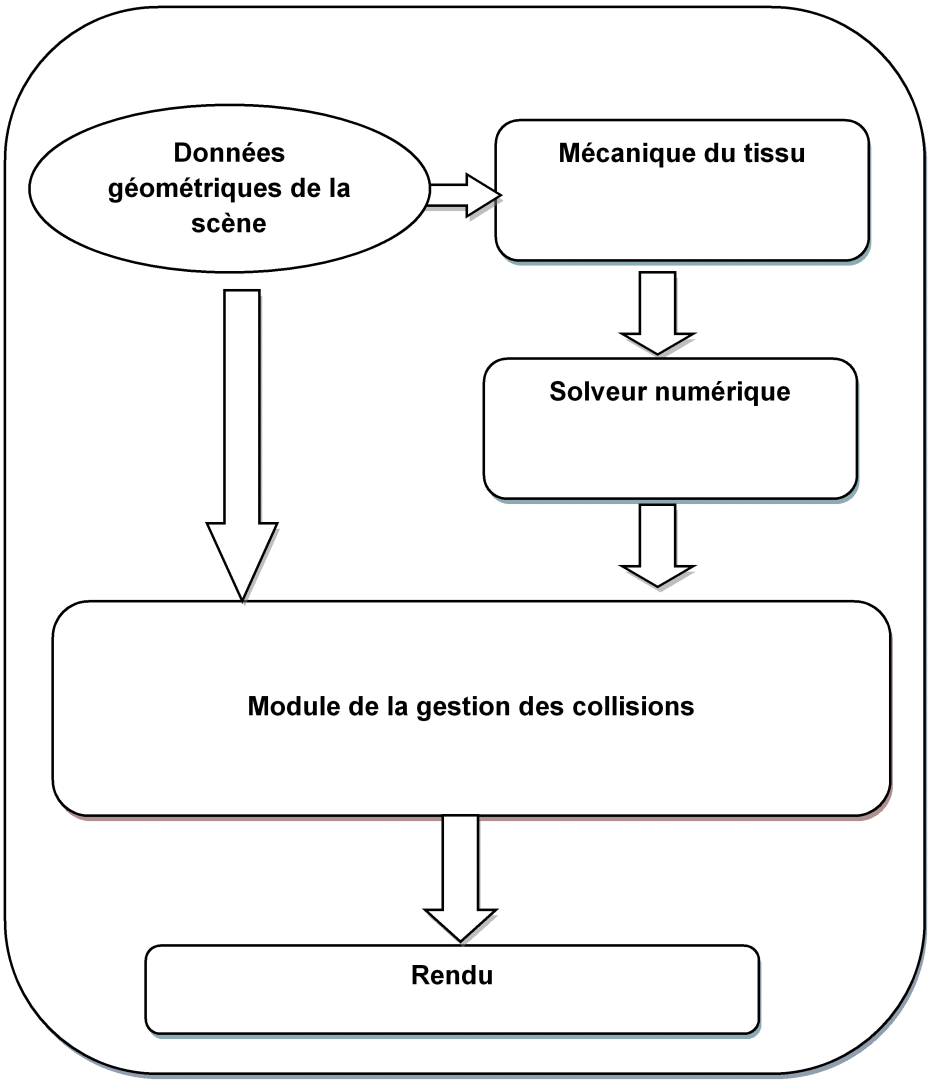


Figure IV.1. Schéma général du processus de simulation du tissu.

2.1. Données géométriques

Ce sont les différentes représentations géométriques des objets de la scène. Elles représentent généralement des maillages polygonaux ou triangulaires. Ces types de données géométriques sont communément utilisés en raison de leur flexibilité.

2.2. Mécanique du tissu

Pour simuler le comportement mécanique du tissu, des forces internes et externes sont prises en compte. Les propriétés physiques du matériau et les forces internes du tissu sont affectées aux maillages en utilisant différentes approches. Les méthodes discrètes comme le

système masse-ressort et le système de particules ont été largement utilisés pour modéliser ces forces internes, qui sont la tension, la force de cisaillement, de flexion et les contractions.

Une autre manière de calculer ces forces consiste en l'utilisation des modèles basés sur la mécanique continue, utilisant les éléments finis. Ces méthodes donnent des performances similaires à celles des autres méthodes avec une qualité bien meilleure.

2.3. Solveur numérique

Additivement aux forces internes, diverses forces externes sont prises en compte, comme la gravité, la résistance de l'air ainsi que les effets du vent. Basée sur les forces internes et externes, l'équation du mouvement du tissu est mise en place en utilisant la deuxième loi de Newton $F=ma$. Pour résoudre l'équation différentielle émergente et obtenir les étapes de simulation discrète, plusieurs types de solveurs numériques ont été utilisés.

2.4. Étape du rendu

Le tissu est visualisé avec son environnement en prenant en compte les propriétés spécifiques de la surface ainsi que les conditions d'éclairage. Ainsi, les propriétés matérielles comme les paramètres de réfraction et de réflexion peuvent être appliquées pour obtenir une apparence réaliste du tissu.

2.5. Module de gestion des collisions

Nous avons laissé ce module en dernier, car c'est à ce niveau que notre travail se focalise. Une tâche essentielle dans le processus de simulation est la réalisation d'interactions entre différents objets. Les interactions les plus importantes sont les collisions entre les objets de la scène. Comme il est nécessaire de faire des tests de détection de collisions entre toutes les primitives des maillages, cette tâche s'avère un goulot d'étranglement pour le processus de simulation. Divers aspects compliquent le problème des collisions pour les objets déformables comme le grand nombre de collisions potentielles et la nécessité de systèmes de traitement des collisions plus complexes. La réponse aux collisions est la deuxième étape de la gestion des collisions, qui résout les collisions détectées et les proximités. Ces méthodes séparent les objets en collision, par exemple en altérant leurs positions ou leurs vitesses. Le processus de gestion de collisions sera représenté ci-dessous.

Le composant détection de collision reçoit en entrée les données géométriques de la scène qui sont représentées par les maillages des objets, et comme sortie une liste des primitives en collision. Cette liste représente l'entrée du composant traitement de collisions.

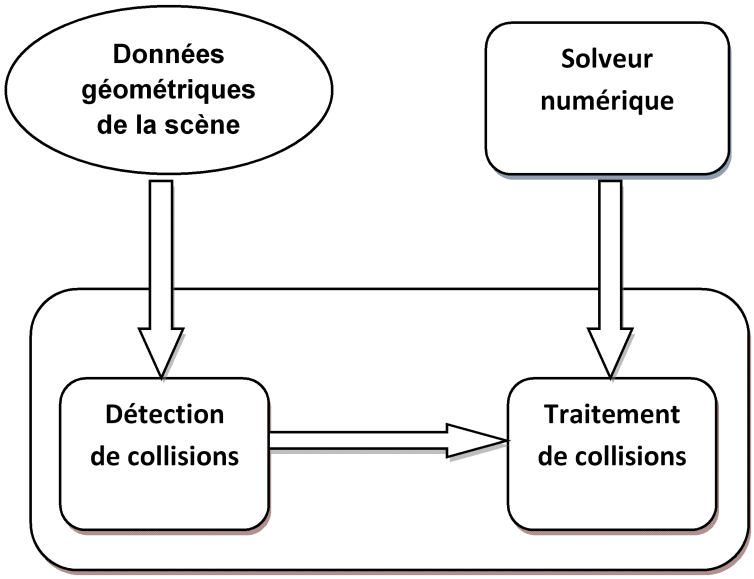


Figure IV.2. Schéma général du processus de gestion des collisions.

2.5.1. Détection des collisions

Notre contribution se situe exactement au niveau de ce composant, puisque nous utilisons les méthodes accélératrices ainsi que la multi résolution pour diminuer le nombre de tests établis et augmenter le niveau de détail dans les zones à risque où des collisions potentielles ont été détectées. Ceci permet d'améliorer la simulation. Ce composant est séparé essentiellement en trois phases, la première est l'application d'une méthode accélératrice, dans notre cas c'est la méthode de la hiérarchie des volumes englobants. La deuxième phase est celle de la multi résolution. Ainsi, une fois une collision détectée, un raffinement sera appliqué sur les primitives en collision pour plus de précision. La dernière étape est la détection exacte et qui consiste en des calculs d'intersections entre les primitives concernées. Une liste des primitives en collision exacte et les informations qui leur sont associées sont transmises en tant que sortie vers le composant de traitement de collisions. Le composant de détection de collisions est représenté par la figure ci-dessous.

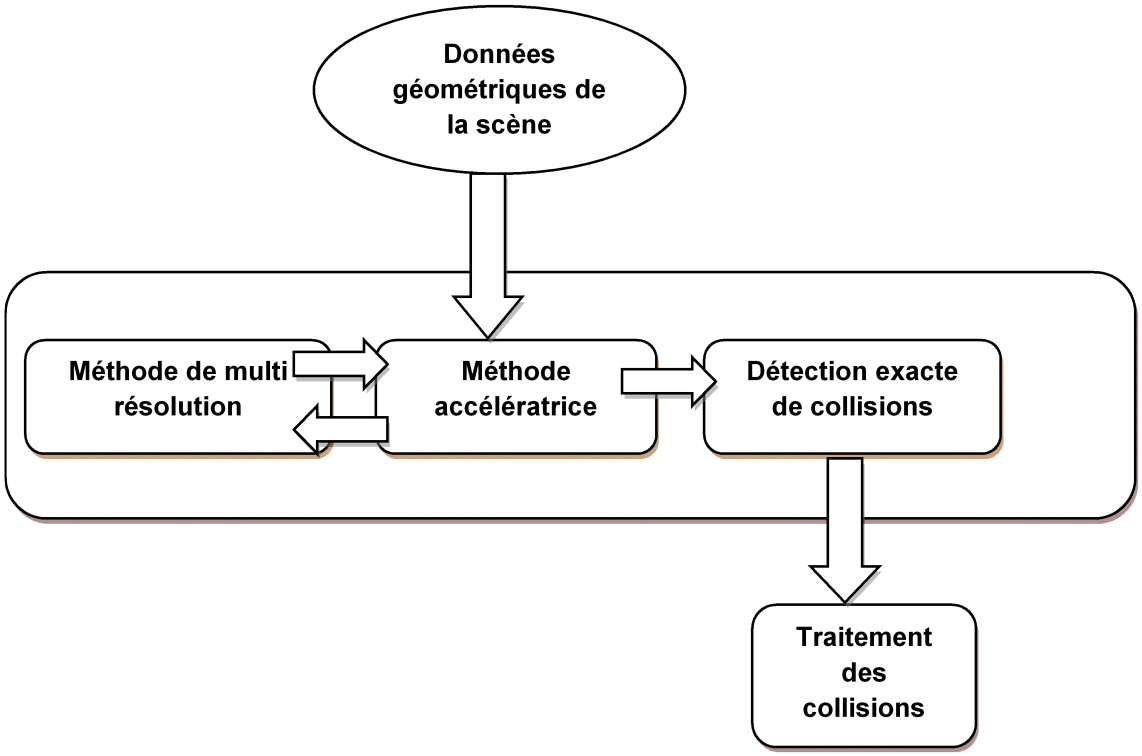


Figure IV.3. Présentation du composant détection de collisions.

2.5.2. Traitement des collisions

Parmi les méthodes proposées pour le traitement des collisions, nous avons choisi la méthode des contraintes basée sur les impulsions. Elle consiste à appliquer des impulsions sur les zones de collision pour éviter les interpénétrations. Cette technique utilise comme entrée la liste des primitives en collision munie des informations concernant cette dernière en plus des données issues du solveur numérique.

2.5.3. Méthode accélératrice

La méthode accélératrice utilisée est la hiérarchie des volumes englobants. Comme volumes nous avons utilisé les sphères pour leur compromis entre approximation et facilité des mises à jour, ainsi que la simplicité du test d'intersection entre sphères. Pour sa construction, nous avons choisi une méthode qui consiste à appliquer récursivement l'algorithme de Welzl sur un maillage triangulaire représenté par ses sommets, qui consiste à construire la plus petite sphère englobant cet ensemble et l'algorithme de subdivision d'un ensemble de points. La hiérarchie sera ensuite parcourue et testée pour d'éventuelles collisions.

3. Hiérarchie des volumes englobants

La hiérarchie des volumes englobants (BVHS) s'est révélée être parmi les structures de données les plus efficaces pour la détection de collisions.

Habituellement, une BVH est construite pour chaque objet dans une étape de prétraitement. L'idée des BVHs est de partitionner récursivement l'ensemble des primitives de l'objet jusqu'à ce qu'un certain critère de feuille soit rempli. Le plus souvent, ce critère est rempli lorsque chaque feuille contient une seule primitive, mais on pourrait aussi s'arrêter quand un nœud contient moins d'un nombre fixe (seuil) de primitives. En général, les BVHs sont définis comme suit: chaque nœud dans l'arbre est associé à un sous-ensemble de primitives de l'objet, entourées d'un volume englobant appartenant à une classe spécifiée de formes [56]. Le choix de cette classe de forme varie entre les AABB, les OBB, les Sphères, les K-DOPs et les enveloppes convexes. Dans notre travail, nous avons choisi les sphères englobantes à cause de leur simplicité de vérification des collisions. Les sphères contiennent juste des informations concernant le rayon et le centre. Ceci facilite les mises à jour de la hiérarchie, qui sont obligatoires pour les objets déformables et qui participent à la lenteur de la simulation. De plus, les sphères englobantes produisent un faible coût de mise à jour.

3.1. Degré de l'arbre

Une BVH est un arbre, une BVH (S) spécifie une hiérarchie de volumes englobants sur S, tel que S est l'ensemble de primitives construisant l'objet en collision, et qui est dans notre cas représenté par un ensemble de triangles. Chaque nœud *v* de la BVH (S) correspond à un sous-ensemble, $S_v \subseteq S$, avec le nœud racine étant associé à l'ensemble S. chaque nœud interne de la BVH (S) a deux enfants ou plus, le nombre maximum d'enfants pour tout nœud interne du BVH (S) est appelé le degré de BVH(S), notée par δ.

Minimiser la hauteur de l'arbre est généralement une qualité souhaitable lors de la construction d'une hiérarchie. Généralement, plus le degré δ est plus élevé, plus la hauteur de l'arbre est réduite. Avec δ qui symbolise le nombre maximum d'enfants pour un seul nœud. De toute évidence, il existe un compromis entre les arbres de haut et faible degré. Un arbre avec un degré élevé aura tendance à être plus court, mais plus d'effort sera demandé pendant la recherche pour chaque nœud. D'autre part, un arbre à faible degré aura plus de hauteur, mais moins d'effort sera nécessaire pendant la recherche pour chaque nœud [89].

Dans ce travail, nous avons choisi d'utiliser les arbres binaires ($\delta = 2$), car ils sont plus simples et plus rapides à calculer, puisqu' il y a moins d'options dans la manière dont on divise un ensemble de primitives en deux sous-ensembles, que de le diviser en trois sous-ensembles ou plus.

3.2. Construction de la hiérarchie des sphères englobantes

Une seule sphère n'approxime pas la forme d'un objet, une union de plusieurs sphères chevauchées et structurées sous forme d'une hiérarchie peut donner une meilleure approximation de l'objet.

La hiérarchie des sphères englobantes est considérée comme un arbre. Chaque niveau de cet arbre contient une union de sphères englobantes un ensemble de primitives de l'objet et représentant un certain niveau d'approximation. L'arbre sera en premier lieu généré dans une étape de prétraitement, puis il sera mis à jour au cours de la déformation de l'objet.

Il existe trois différentes stratégies pour construire les BVHs, à savoir top-down, bottom-up et la stratégie d'insertion. Cependant, la stratégie top-down est la plus couramment utilisée pour la détection de collisions.

L'idée est de diviser récursivement un ensemble de primitives de l'objet jusqu'à ce qu'un seuil spécifié soit atteint. Le fractionnement est guidé par un critère spécifié par l'utilisateur ou une heuristique ce qui donnera une BVH respectant le critère choisi [56].

Les exigences essentielles pour construire une hiérarchie de sphères englobantes sont précisées par Hubbard [88] est qui sont :

- La phase de prétraitement doit être automatique, sans avoir besoin d'intervention de l'utilisateur.
- La hiérarchie doit être structurée de telle manière à rendre la recherche efficace, où à chaque niveau il faut éliminer le besoin de rechercher un sous-ensemble significatif de niveau supérieur.
- Chaque hiérarchie doit approximer l'objet qu'elle représente de la façon la plus adéquate.

Pour la construction de notre arbre, plusieurs décisions de conception doivent être prises. Au début, nous cherchons à englober tous les éléments du maillage avec une seule sphère qui représentera le premier nœud de l'arbre. Cette sphère sera construite en respectant la méthode de Welzl pour le calcul de la sphère minimale englobant un ensemble d'éléments. Cette

méthode sera utilisée à chaque niveau de la hiérarchie, pour la construction des sphères englobant les ensembles d'éléments issus de chaque subdivision de l'ensemble du nœud père.

Par la suite, le volume englobant du premier niveau de la hiérarchie doit être subdivisé en deux, puisque nous travaillons avec les arbres binaires. Sachant que *v* est un nœud de l'arbre *BV* correspondant à l'ensemble des primitives de l'objet S_v, et que *v'* est le nœud enfant du nœud *v*. L'assignement des primitives de l'objet à chaque nœud enfant correspond au partitionnement de l'ensemble S_v en deux. Ceci représentera $\frac{1}{2}(2^{|Sv|}$ -2) manière de faire ce partitionnement. Par conséquent, nous ne pouvons pas considérer tous les partitionnements possibles. Ceci nous permet d'associer chaque triangle de l'ensemble S_v à un seul point représentatif qui sera le centre, et nous subdivisons S_v en deux parties, par rapport à un plan orthogonal sur l'un des trois axes de coordonnées, et en attribuant un triangle sur le côté du plan où se trouve le centre de gravité. Cela résulte d'environs 3*($|Sv|$-1) différentes subdivisions, puisqu'il y a trois choix de l'axe, et pour chaque axe il y a $|Sv|$-1 subdivisions différentes pour chacun des points représentant les centres de gravité [89].

3.2.1. Choix de l'axe

Nous choisissons un plan orthogonal à l'un des axes x, y, ou z en nous basant sur les fonctions objectives suivantes :

Min sum : choisir l'axe qui minimise la somme des volumes des deux nœuds enfants qui en résultent.

Min max : choisir l'axe qui minimise le plus grand des volumes des deux nœuds enfants issus.

Largest variance : Projeter les centres des triangles sur chacun des trois axes de coordonnées et calculer la variance de chacune des distributions résultantes. Choisir l'axe donnant la plus grande variance.

Dans notre travail nous avons choisi d'utiliser la fonction de covariance puisqu'elle est plus rapide que les deux autres, vu qu'elle se déroule en temps linéaire O ($|S_v|$). Ainsi, chaque primitive de l'objet sera approximée par son centre. Après, pour un ensemble de points (les centres approximant les primitives de l'objet) *B* leurs principales composantes (les vecteurs propres de la matrice de covariance) sont calculées, et le plus grand d'entre eux c.-à-d. celui ayant la plus grande variance est choisi. Par la suite, un plan orthogonal à cet axe principal sera placé à travers le point de subdivision choisi.

3.2.2. Choix du point de subdivision

Une fois l'axe choisi, celui-ci étant orthogonal au plan de subdivision, nous devons choisir la position du plan diviseur parmi les ($|Sv|$-1) possibilités. Il y a généralement deux manières de le faire: soit en utilisant la moyenne des coordonnées du centre selon l'axe choisi, soit la médiane.

Ainsi, après chaque subdivision, nous construisons les sphères englobantes ainsi que les ensembles de primitives représentés par les nœuds enfants qui en résultent. La construction des sphères se fait en utilisant l'algorithme de Welzl pour le calcul de la sphère minimale englobant un ensemble d'éléments. Par la suite, ce processus d'opérations se doit se répéter au cours de la construction de chaque niveau de la hiérarchie.

3.3. Algorithme de Welzl

C'est un algorithme aléatoire qui calcule le plus petit disque englobant d'un nombre fini de points dans le plan en temps linéaire.

Nous avons un ensemble P de n points dans le plan, nous voulons calculer le disque fermé du plus petit rayon D (P) qui contient tous les points de P. Supposons que nous avons déjà un SED (Smallest Enclosing Disk) D, pour (n-1) points $p_1 \ldots p_{n-1}$. Ainsi, on doit considérer deux cas pour le $n^{ème}$ point :

1) p_n se trouve à l'intérieur de D: donc le SED D pour $p_1,..., p_{n-1}$ est le même pour $p_1,..., p_n$.

2) p_n se trouve à l'extérieur de D: donc, nous devons calculer un nouveau SED, mais nous savons que n doit se trouver sur la frontière de D. donc nous allons calculer un SED D' de $p_1 \ldots . p_{n-1}$ avec p_n sur la frontière de D'.

Nous pouvons désormais calculer le SED d'une manière itérative en nous basant sur les trois lemmes suivants :

1) S'il existe un disque contenant P avec R (tel que R est l'ensemble des points se situant sur la frontière) sur sa frontière, alors D (P, R) est bien défini et unique.

2) Si p ne se trouve pas à l'intérieur de D (P - {p}, R), alors p se trouve sur la frontière de D (P, R), à condition qu'il existe, ce qui signifie que :

D (P, R) = D (P - {p}, R u {p}).

3) Si D (P, R) existe: il y a un ensemble S d'au maximum {0, 3 - | R |} points dans P tels que :

D (P, R) = D (S, R).

Cela signifie que P est déterminé par au plus 3 points de P qui se trouvent sur la frontière de D (P) [92].

Le pseudo-code suivant représente le calcul du SED d'un ensemble de points [92]:

```
function SED(P,R)
{
if (P is empty or |R| = 3)   then D: = calcDiskDirectly(R);
                             else choose a p from P randomly;

D: = SED(P - {p}, R);
if (p lies NOT inside D) then D: = SED(P - {p}, R u {p});
return D;
}
```

3.4. Parcours de la hiérarchie

Pour le test de collision, la hiérarchie des volumes englobants est parcourue de haut en bas. Et les paires de nœuds de l'arbre sont récursivement testées s'ils sont en collision. Si les nœuds chevauchés sont des feuilles, alors les primitives englobées sont testées pour l'intersection. Si un nœud est une feuille et l'autre est un nœud interne, alors le nœud feuille est testé avec chaque enfant du nœud interne. Si les deux nœuds sont des nœuds internes alors, il faut tenter de minimiser la probabilité d'intersection le plus vite possible. Par conséquent, on teste le nœud avec le plus petit volume contre les enfants du nœud avec le plus grand volume [56]. Pour deux objets donnés avec une BVH A et B, la plupart des algorithmes de détection de collisions mettent en œuvre le schéma général de l'algorithme suivant[60]:

```
traverse (A,B)
if A and B do not overlap then
return
end if
if A and B are leaves then
return intersection of primitives enclosed by A and B
else
for all children A[i] and B[j]do
traverse(A[i],B[j])
end for
end if
```

3.5. Mis à jour de la hiérarchie

Contrairement aux hiérarchies pour les objets rigides, où l'objet complet subit une transformation, les hiérarchies pour les objets déformables doivent être mises à jour à chaque pas de temps. Principalement, il y a deux possibilités pour cette mise à jour: le réaménagement et la reconstruction. Le réaménagement adapte les BV aux nouvelles

positions des primitives englobées, tandis que la reconstruction reconstruit complètement la BVH. Le réaménagement est beaucoup plus rapide que la reconstruction, mais pour les grandes déformations, ceci conduit à une mauvaise approximation des volumes englobants, et par la suite à plus de chevauchements [60].

Pour la mise à jour de notre hiérarchie, nous avons choisi la méthode appliquée dans les travaux de Van der Bergen [84], en réajustant cette dernière en utilisant les sphères englobantes au lieu des AABBs. Après une déformation qui représente un changement des emplacements et des formes des primitives du modèle par rapport au système de coordonnées locales du modèle au cours du temps, une première étape consiste à recalculer les sphères englobantes des feuilles. Ensuite, chaque sphère mère est recalculée en utilisant les sphères de ses enfants dans un ordre strictement ascendant c.-à-d. de bas en haut. Par ailleurs, les feuilles et les nœuds internes de la hiérarchie sont représentés par un tableau de nœuds. En outre, l'arbre est construit de sorte que l'index de chaque nœud enfant d'un nœud interne dans le tableau est supérieur à l'index de son parent. De cette manière, les nœuds internes sont parcourus en itération dans le tableau des nœuds internes dans un ordre renversé. En d'autres termes, le but de la méthode est d'utiliser un tableau de nœuds dans lequel on s'assure que l'indice d'un nœud est toujours plus grand que celui de son père. Ainsi, le réajustement se fait en parcourant ce tableau dans l'ordre inverse.

Les mises à jour des hiérarchies des sphères entraînent le changement des centres et des rayons de ces sphères, ce qui cause la perte d'exactitude. Communément, la nouvelle position du centre est déterminée en ajoutant à l'ancien centre le vecteur de déplacement des sommets inclus. Le nouveau rayon est obtenu en calculant la distance du nouveau centre au point le plus loin, ce qui affecte l'exactitude de la hiérarchie. Par conséquent, il est nécessaire de garder un niveau approprié d'exactitude en essayant de réduire l'incrémentation excessive du rayon.

3.5.1. Mise à jour du centre

Pour chaque facette d'intersection F_i d'une sphère donnée *s*, on obtient les sommets v_j tels que $v_j \in F_i$ et associé à *s*. Soit *BB* le volume englobant de l'ensemble des sommets v_j associé à *s* avec c_{BB} est le centre de *BB*. Soit *u* le vecteur de déplacement du centre du volume englobant *BB*, donc [91] :

$$u = C_{BB\ new} - C_{BB\ original}.$$

Par conséquent, la nouvelle position du centre est donnée par :

$$c_{new} = c + u$$

Ce processus est répété pour toutes les sphères dans la hiérarchie.

3.5.2. Mise à jour du rayon

Soit d_j la distance entre $v_j \in s$ et le centre de la sphère c dans la configuration initiale. Soit $d_{j\,new}$ la distance entre les nouvelles positions de v_j et le nouveau centre. On définit le rapport d'incrémentation par [91] :

$$\alpha_j = d_{j\,new} / d_j$$

Ainsi, pour une sphère feuille, le nouveau rayon r_n est calculé comme suit :

$$r_n = r\ max\ (\alpha)$$

Où r est le rayon original, ceci se fait de manière hiérarchique. Ainsi, pour des niveaux supérieurs, au lieu d'utiliser les sommets, nous employons la distance à ses sphères enfants.

Soit d_{cj} la distance entre les centres de s et celles des sphères enfants associées s_j, et r_j le rayon original de s_j. Pour une sphère interne d_j est calculée comme suit :

$$d_j = dc_j + r_j$$

Par conséquent si $dc_{j\,new}$ est la distance entre le nouveau centre de s et le nouveau centre de s_j et $r_{j\,new}$ est le nouveau rayon de s_j alors $d_{j\,new}$ est calculée comme suit :

$$d_{j\,new} = dc_{j\,new} + r_{j\,new}$$

Par conséquent, le rayon sera calculé comme suit :

$$r_n = r\ max(\alpha)$$

Le pseudo-code suivant représente le processus de mise à jour de l'arbre [94]:

```
{filsDeformés : liste des nœuds d'un niveau devant être mis à jour}
{pèresDeformés : liste des nœuds du niveau suivant qui devront être
mis à jour}
pèresDeformés.vide()
```

```
while !filsDeformés.estVide() do
for all noeuddansfilsDeformédo
ifnoeud.nécessiteMiseAJourthen
mettreAJourSphère(noeud)
noeud.nécessiteMiseAJour = faux
ifnoeud.pere != NUL then
noeud.pere.nécessiteMiseAJour = true
pèresDeformés.ajouteEnFin(noeud.pere)
end if
end if
end for
filsDeformés = pèresDeformés
pèresDeformés.vide()
end while
```

4. Association de la multi résolution à la hiérarchie des volumes englobants

Le but de l'algorithme de gestion des collisions avec la multi résolution est d'améliorer la performance de détection et réponse aux collisions en utilisant des représentations triangulaires adaptatives se rapprochent du maillage original. L'adoption d'un algorithme de multi résolution pour la simulation de vêtements vise à optimiser les performances de temps de calcul en représentant seulement les parties les plus importantes ou réellement actives du modèle du tissu avec une résolution de haut niveau [93].

Dans notre travail, nous avons cherché à intégrer la multi résolution pour la simplification de la détection de collisions. L'idée consiste à générer une série de représentations qui s'étalent entre la représentation la plus détaillée jusqu'à la plus simplifiée par rapport aux nombres de triangles formant les maillages des objets en collision.

Pour garder le système fiable, il est utile d'utiliser une représentation appropriée dépendante de l'état dans lequel se trouvent les objets. Sachant que la hiérarchie des sphères englobantes est déjà construite avant le début de la simulation, une représentation simplifiée lui est associée. Au fur et à mesure de la réalisation de la simulation, notre scène simulée variera entre les différentes représentations, et suivant chaque représentation une hiérarchie de sphères englobantes est construite. Ainsi, l'algorithme du parcours de la hiérarchie traverse l'arbre de l'objet A et celui de l'objet B en vérifiant l'existence d'intersections entre une paire de sphères a et b appartenant respectivement à l'arbre A et à l'arbre B. S'il n'y a pas

d'intersection, l'algorithme de parcours et de test s'arrête, et ainsi une représentation simplifiée des deux objets est utilisée.

Par contre, si a et b se croisent alors une représentation plus détaillée des objets est utilisée et l'algorithme de parcours et de test est appliqué récursivement à leurs enfants. Si a et b sont deux nœuds feuilles, les primitives les composants sont testées directement et les objets sont représentés par la représentation la plus détaillée.

L'efficacité de l'intégration de la multi résolution dans le processus de détection de collisions se base sur deux points essentiels, le premier consiste à créer des représentations multi résolutions précises. Le deuxième concerne l'intégration de ces différentes représentations dans une hiérarchie efficace de volumes englobants, et ce d'une manière adaptative.

4.1. Représentations multi résolution

Nous avons appliqué la technique de multi résolution dans notre travail en nous inspirant de la méthode de Hutchinson [86] qui permet le raffinement adaptatif des systèmes masse/ressort. Ce procédé consiste à calculer, pour un grillage initial de m × n carrés, les forces exercées par les ressorts sur les masses ainsi que les forces extérieures à chaque pas de la simulation. On en déduit les accélérations puis l'état au prochain pas (position et vitesse des masses du système). Au cours de la simulation, lorsque deux arêtes contiguës (et opposées) forment un angle trop grossier, chacun des 4 carrés autour du point correspondant (entre les deux arêtes) est divisé en 4. Toutes les masses faisant partie des 4 carrés sont d'abord remises à l'état précédent, soit au moment où leur état était encore correct. Ensuite la subdivision est effectuée, puis les positions des masses sont recalculées avec la nouvelle configuration géométrique de la zone.

Nous avons essayé d'adapter ce procédé à notre travail en l'appliquant sur un système de masse ressort triangulé [45], en remplaçant sa méthode de subdivision par une méthode d'activation des particules qui sera détaillée plus loin. Le critère de raffinement représente la détection d'une collision potentielle. De plus, nous avons associé ce processus à celui du parcours et test de la hiérarchie des sphères englobantes pour pouvoir réaliser notre approche.

4.2. Critères à respecter

Pour développer sa méthode, Hutchinson a défini trois propriétés que le système doit toujours remplir pour avoir un comportement cohérent. Ces critères se basent, notamment, sur les caractéristiques des systèmes masse-ressort statiques, composés de masses et de ressorts identiques [86] :

- Les particules du système doivent se comporter de manière identique par rapport aux forces ;
- La répartition des masses doit être homogène; la somme des masses sur une certaine zone doit rester constante, notamment après l'ajout de nouveaux points ;
- La vitesse de propagation des forces à travers le tissu doit être identique partout sinon la simulation devient incohérente.

4.3. Présentation de la méthode

Généralement, les représentations multi résolution sont souvent créées en décimant les modèles polyédriques donnés. Les difficultés surgissent lorsqu'on tente d'intégrer ces représentations dans des BVHs. Considérant chaque LOD d'un objet donné comme étant un modèle dans son ensemble, chaque niveau de détail (LOD) exigerait une BVH distincte pour la détection de collisions [87].

Soient :M_0 : le maillage triangulaire initial.

$\{M_0, M_1,..., M_{n-1}\}$: une séquence de niveau de détails.

c_i : un ensemble de triangle.

C_i : le volume (sphère) englobant l'ensemble de triangles c_i.

Le niveau de détails (LOD) M_j de résolution r_j d'un maillageM_0 peut être obtenu à partir d'un LOD M_i de résolution r_i plus élevée en supprimant des détails à des résolutions dans l'intervalle $[r_j, r_i]$ si M_0 est considéré comme le niveau de détails avec la résolution la plus élevé.

Le niveau de détails (LOD) M_j de résolution r_j d'un maillage M_0 peut être obtenu à partir d'un LOD M_i de résolution r_i plus faible en ajoutant des détails à des résolutions dans

l'intervalle [r_j ,r_i] si M_0 est considéré comme le niveau de détails avec la résolution la plus faible.

Dans notre système, le tissu est construit selon le niveau de raffinement le plus détaillé, pour conserver nos particules dans une structure de données. Ceci permet de pouvoir accéder facilement à une particule donnée dans n'importe quel niveau de raffinement.

Le tissu, à un niveau de raffinement donné, peut être représenté par un tableau où chaque particule du maillage du tissu possède un index unique dans le tableau.

A ce niveau, nous avons un tableau de particules dans tous les niveaux de raffinement. Ceci signifie que nous avons un tableau avec sa partie initiale représentant le tissu au niveau le plus grossier, sa partie ultérieure représentant le tissu au niveau suivant et ainsi de suite. Cette approche est gourmande en termes d'utilisation mémoire parce que nous avons des copies multiples des particules au niveau inférieur ainsi qu'au niveau supérieur. Cependant, vu que nous travaillons seulement avec trois niveaux de raffinement, ceci ne représente pas un inconvénient paralysant.

Puisque nous connaissons le maximum de particules à chaque raffinement, et étant donné que toute particule possède un index dans le tableau, nous pouvons facilement accéder à sa représentation dans les autres niveaux. Chaque particule du maillage représentant le tissu ne peut être marquée active dans le tableau que dans un seul niveau de raffinement à un instant t.

On maintient également un tableau de particules actives qui stocke les indices de particules actives dans le tableau de toutes les particules est essentiel pour un accès facile à la particule actuellement active dans le tissu.

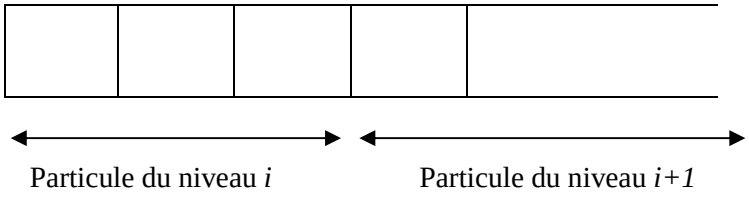


Figure IV.4 Tableau des particules du tissu.

4.4. Intégration de la multi résolution

L'adoption d'un algorithme de multi résolution pour la simulation de tissu vise à optimiser la performance de la simulation en réduisant le temps d'exécution. Ceci se fait en ne représentant, à une résolution plus élevée, que les parties du modèle de tissu qui sont considérées comme critiques, ou sujettes à des collisions.

Cependant, durant l'exécution, quand la simulation est en train de se dérouler, certaines parties du maillage peuvent gagner en détail et deviennent plus complexes, tandis que d'autres peuvent en perdre et deviennent moins complexes. Nous raffinons ou simplifions les parties du maillage de manière adaptative selon les exigences de la résolution de ces parties.

Ainsi, nous maintenons la résolution initiale du maillage au cours de la simulation, jusqu'à aboutir à la détection d'une collision potentielle, par l'algorithme de parcours et de test de la hiérarchie des volumes englobants. Une fois les feuilles repérées par une collision, un raffinement à ce niveau est effectué utilisant l'algorithme de l'activation des particules.

Le tableau de particules du tissu est rempli avec toutes les particules du maillage raffiné suivant l'ordre de leur création, ensuite nous appliquons la méthode d'activation des particules qui nous permet d'activer les particules concernées par le niveau de raffinement grossier, après avoir fixé le nombre de particules du maillage initial. Ceci nous permet de construire le tissu avec une représentation grossière au début de la simulation.

En association avec le tableau qui contient toutes les particules existantes, nous en utilisons un autre pour les particules actives à chaque pas de la simulation.

Une fois qu'une collision potentielle est repérée, un raffinement s'effectue en activant les particules adéquates.

Ce raffinement nous permet de valoriser la région de collision en lui ajoutant plus de détails, ce qui conduit à préciser les points de collision et gagner en termes de qualité.

Vu que la simulation produit des déformations et des mouvements des objets simulés, les parties détectées pour des collisions peuvent ne plus être en intersection. Donc le raffinement effectué sur ces parties sera annulé.

4.5. Propriétés physiques

Le maillage du tissu possède plus d'attributs que ceux de la géométrie. Ces attributs dépendent du modèle physique utilisé. Comparé aux maillages ordinaires, celui du tissu est augmenté avec des vecteurs de vitesse, des coordonnées de masse ainsi que différentes forces externes.

4.5.1. Les masses

La méthode ne réduit pas la masse des particules raffinées dans le but de maintenir le premier critère de Hutchinson, qui impose un comportement identique aux particules : si celles-ci ont des masses différentes, elles réagiront de façon différente aux forces. Cette mesure provoque néanmoins la violation du deuxième critère, qui consiste à assurer une densité massique constante sur toute la surface. Pour pallier cette incohérence, on double la rigidité des ressorts à proximité des nouveaux points, ce qui corrige l'interaction des particules entre elles [86].

4.5.2. Vitesse de transmission des forces

Le troisième critère de Hutchinson exige que les forces soient transmises à la même vitesse sur toute la surface. Si la période de mise à jour des particules est la même partout, une force se répandra deux fois moins vite dans une région raffinée que dans une région non raffinée, car les ressorts sont deux fois plus courts. De ce fait, nous devons adapter l'intervalle de temps entre deux mises à jour par rapport au niveau de subdivision de la région.

La période d'intégration originelle T, déterminée par l'utilisateur, est fractionnée pour donner la période effective valant $T/2^n$, où n est le niveau de subdivision actuel maximal. Par exemple, pour une particule de niveau i, les forces appliquées à cette particule sont recalculées tous les $T/2^i$. Entre les deux, les forces restent constantes et la vitesse et la position sont mises à jour en conséquence. Cette réduction du temps d'intégration permet d'assurer une autre fonction importante, celle de permettre d'adapter la période minimale nécessaire à la non-divergence du système [45].

4.5.3. Construction des ressorts

De nouveaux ressorts sont ajoutés au système de sorte qu'ils ne perturbent pas les forces agissant sur les particules du tissu. Deux types de scénarios sont possibles:

- Deux petits ressorts sont formés en divisant un plus grand ressort. La longueur des deux nouveaux ressorts est la moitié de celle du ressort initial. Leurs constantes de rigidité sont deux fois plus grandes que celle du ressort initial.
- Un nouveau ressort est formé, sa longueur et les constantes de rigidité de ce type de ressorts sont assignées par approximation en tenant compte de ne pas perturber la stabilité du système.

5. Collision sphère/sphère

Un des avantages de l'usage des hiérarchies des sphères est les calculs très simples et plus efficaces lors des tests de collisions entre deux sphères, la figure IV.5 montre ces calculs.

Si la somme des deux rayons est inférieure à la longueur l qui représente la distance entre les deux centres alors il n'y a pas de collision. Par contre si la somme est supérieure ou égale à la longueur l alors il existe bien une collision.

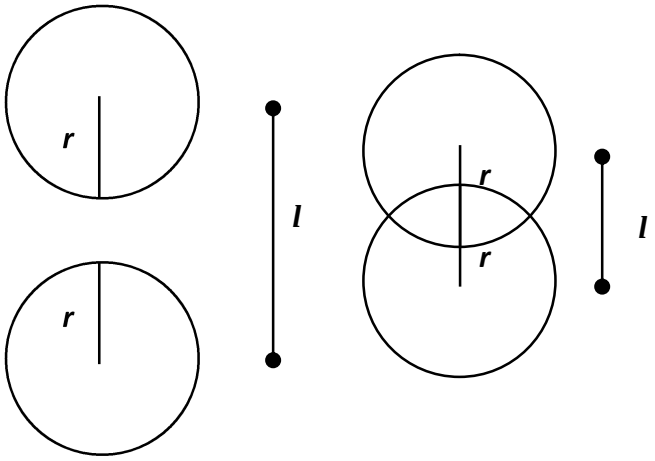


Figure IV.5. Détection de collisions entre deux sphères.

Pour S_1 et S_2 deux sphères de centres respectivement c_1 et c_2 avec les rayons r_1 et r_2, une seule manière d'implémenter ce test c'est de vérifier que [88] :

$$|c_1 - c_2| < r_1 + r_2$$

Cependant, nous avons :

$$|c_1 - c_2| = \sqrt{(c_1 - c_2).(c_1 - c_2)}$$

Donc :

$$(c_1 - c_2).(c_1 - c_2) < (r_1 + r_2)^2$$

Ce qui est un test très rapide pour la vérification.

6. Détection de collision exacte

La hiérarchie des volumes englobants est utilisée comme une étape initiale au processus de détection de collisions. Le résultat de cette étape consiste en un ensemble de paires de triangles qui sont candidats pour la détection de collision. Soit t_i l'instant initial, où les deux maillages triangulaires correspondants ne sont pas en collision. Durant l'intervalle de temps [t_i , t_i +Δt], et qui représente un pas de temps, les sommets de ces deux maillages se déplacent et des collisions entre les triangles du maillage peuvent se produire. Nous supposons que pour chaque sommet, la position et la vitesse initiale à l'instant ti sont données. Les nouvelles positions des sommets à l'instant $t_i + \Delta t$ peuvent être facilement calculées en supposant que le mouvement est uniforme, c'est-à-dire que tous les sommets se déplacent à une vitesse constante. En général, les deux triangles peuvent entrer en collision de deux manières différentes. Soit un sommet du premier triangle entre en collision avec l'autre triangle (figure IV.6.(a)) ou bien deux arêtes de deux triangles différents se croisent (figure IV.6.(b))[79].

a) Collision Vertex/triangle b) Collision arête/arête

Figure IV.6 Les deux types de collisions possibles entre triangles [60].

6.1. Le temps de la collision

Pour détecter les collisions entre un sommet et un triangle ou entre deux arêtes en mouvement, il est nécessaire de trouver l'instant dans l'intervalle [t_i , $t_i + \Delta t$], au cours duquel les quatre sommets impliqués sont coplanaires. Pour une collision entre les deux primitives, cette condition est nécessaire.

Soit $\overrightarrow{v_1}, \overrightarrow{v_2}, \overrightarrow{v_3}, \overrightarrow{v_4}$ les vitesses constantes pendant l'intervalle $[t_i, t_i + \Delta t]$ des quatre points P_1 en mouvement, P_2, P_3 et P_4. Leurs vecteurs de positions respectives sont donnés par:

$$\overrightarrow{x_1}(t) = \overrightarrow{x_1}(t_i) + (t - t_i)\overrightarrow{v_1}$$
$$\overrightarrow{x_2}(t) = \overrightarrow{x_2}(t_i) + (t - t_i)\overrightarrow{v_2}$$
$$\overrightarrow{x_3}(t) = \overrightarrow{x_3}(t_i) + (t - t_i)\overrightarrow{v_3}$$
$$\overrightarrow{x_4}(t) = \overrightarrow{x_4}(t_i) + (t - t_i)\overrightarrow{v_4}$$

Les vecteurs de différence $\overrightarrow{x_1x_2}(t)$, $\overrightarrow{x_1x_3}(t)$ et $\overrightarrow{x_1x_4}(t)$, construisent un volume parallélépipédique. A l'instant $t \in [t_i, t_i + \Delta t]$ pour lequel ce volume est zéro, les quatre points sont coplanaires. Cela se traduit par la recherche des racines de l'équation cubique suivante:

$$(\overrightarrow{x_1x_2}(t) \times \overrightarrow{x_1x_3}(t)) \cdot \overrightarrow{x_1x_4}(t) = 0 \qquad (1)$$

Cette équation donne un maximum de trois racines réelles. Toute racine en dehors de l'intervalle $[t_i, t_i + \Delta t]$ est rejetée. Pour les racines qui restent elles sont vérifiées si elles remplissent la condition suffisante d'entraîner une collision. Si des collisions sont trouvées à plusieurs reprises à l'instant t, seule la première est manipulée, les autres sont rejetées.

6.2. La collision sommet/triangle

Pour une collision entre un sommet et un triangle, la condition suffisante est que le sommet se trouve dans le triangle. Ainsi, d'abord, il est vérifié si le sommet p_4 est plus proche d'une valeur de tolérance ε du plan contenant le triangle$P_1P_2P_3$ avec la normale $\vec{n}$: $|\overrightarrow{x_1x_2}(t) \cdot \vec{n}| < \varepsilon$.

Ensuite, le sommet P_4 est projeté sur ce plan et les coordonnées barycentriques w_1, w_2, w_3 par rapport au triangle sont calculées en utilisant l'équation(2). Si toutes les coordonnées barycentriques se trouvent dans l'intervalle [0, 1] et si $w_1 + w_2 + w_3 = 0$, alors une collision se produit.

$$\begin{pmatrix} \overrightarrow{x_1x_3}(t) \cdot \overrightarrow{x_1x_3}(t) & \overrightarrow{x_1x_3}(t) \cdot \overrightarrow{x_2x_3}(t) \\ \overrightarrow{x_1x_3}(t) \cdot \overrightarrow{x_2x_3}(t) & \overrightarrow{x_2x_3}(t) \cdot \overrightarrow{x_2x_3}(t) \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} \overrightarrow{x_1x_3}(t) \cdot \overrightarrow{x_3x_4}(t) \\ \overrightarrow{x_2x_3}(t) \cdot \overrightarrow{x_4x_3}(t) \end{pmatrix} \qquad (2)$$

La valeur de tolérance ε peut être utilisée pour prendre en compte les erreurs d'arrondi ou pour prendre en compte l'épaisseur du tissu.

6.3. La collision arête/arête

Pour détecter les collisions entre une paire d'arêtes qui respectent l'équation (1) à un instant précis *t*, les deux points, un sur chacune des deux arêtes $\overline{P_1P_2}$ et $\overline{P_3P_4}$ qui sont les plus proches sont trouvés et leur distance euclidienne est vérifiée. Si cette distance est inférieure à la valeur de tolérance spécifiée δ, les deux arêtes sont en collision à l'instant t.

La recherche pour les deux points les plus proches commence par vérifier le cas dégénéré d'arêtes parallèles, à savoir si le produit vectoriel $|\overrightarrow{x_1x_2}(t) \times \overrightarrow{x_3x_4}(t)|$ est plus petit qu'une tolérance d'arrondi. Sinon, les points alignés sur les lignes infinies obliques et qui sont proches les unes aux autres sont trouvés en résolvant l'équation (3).

$$\begin{pmatrix} \overrightarrow{x_1x_2}(t)\cdot\overrightarrow{x_1x_2}(t) & -\overrightarrow{x_1x_2}(t)\cdot\overrightarrow{x_3x_4}(t) \\ -\overrightarrow{x_1x_2}(t)\cdot\overrightarrow{x_3x_4}(t) & \overrightarrow{x_3x_4}(t)\cdot\overrightarrow{x_3x_4}(t) \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \overrightarrow{x_1x_2}(t)\cdot\overrightarrow{x_1x_3}(t) \\ -\overrightarrow{x_3x_4}(t)\cdot\overrightarrow{x_1x_3}(t) \end{pmatrix} \quad (3)$$

Si les points résultants avec des vecteurs de position $x_1(t) + a\overrightarrow{x_1x_2}(t)$ et $x_3(t) + b\overrightarrow{x_3x_4}(t)$ sont sur les arêtes finies, les points avec la plus petite distance se trouvent et le processus est terminé. Si ces deux points ne sont pas sur les arêtes finies, ils sont déplacés vers les plus proches extrémités des arêtes. Le point qui se déplace le plus durant ce processus est l'un des deux points avec la plus petite distance. Le second point peut être trouvé en projetant le premier sur la seconde ligne à l'infini et si ce point n'est pas aligné sur l'arête finie, le déplacer vers la plus proche extrémité [60].

7. Traitement de collisions

La réponse aux collisions vise à reproduire les interactions entre les objets d'une façon précise. Elle consiste à trouver les nouvelles positions et vitesses des objets en collision et ceci dans le but d'éviter les interpénétrations irréalistes des objets. En pratique, il existe deux classes de méthodes, celle basée sur les pénalités et celle basée sur les contraintes. La première consiste à détecter toutes les collisions, et de placer un ressort virtuel de longueur à vide nulle au niveau de chaque collision. Ce dernier va induire une force répulsive qui va avoir tendance à faire éloigner les deux objets l'un de l'autre. La force générée dépend de la

valeur de l'interpénétration fictive entre les deux objets. La deuxième 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. Ceci est fait en modifiant les forces du tissu afin d'éviter sa collision avec l'objet [85].

Dans notre travail, nous avons choisi d'utiliser la méthode des contraintes, parce qu'elle offre des réponses précises aux collisions sans l'utilisation de forces de grandes valeurs.

Cette méthode a été appliquée dans plusieurs travaux parmi eux, le travail de Kimmerl dans [60] dont nous allons reprendre la partie concernant la réponse aux collisions pour le système de simulation de vêtements TuTex, cette méthode est basée sur les impulsions.

L'idée derrière la méthode des impulsions est de considérer la collision entre les objets comme un impact, où les impulsions dans des directions opposées sont appliquées sur les points du maillage concernés afin d'éviter une interpénétration. Une approximation du comportement complexe lors de la collision d'objets avec la déformation et la restitution des formes est donnée par l'hypothèse de Poisson. L'utilisation de l'hypothèse de Poisson pour les collisions du tissu, le paramètre du matériau pour la restitution est normalement supposé être de zéro, précisant qu'aucune énergie n'est stockée dans la déformation pendant le processus de collision. Ainsi, les interactions d'objets en tissu sont modélisées en supposant que les collisions sont totalement inélastiques.

Si deux points à l'intérieur des primitives du maillage en collision s'approchent l'un de l'autre avec une vitesse relative dans la direction de la normale $\overrightarrow{v_{relN}}$ et sont plus proches de la distance de collision, une impulsion est appliquée afin d'éviter l'interpénétration. Cette impulsion nécessaire ainsi que la vitesse des points à l'intérieur des primitives sont calculées en utilisant les relations suivantes :

$$\overrightarrow{v_{relN}} = 0 = \overrightarrow{v_4} - w_1\overrightarrow{v_1} - w_2\overrightarrow{v_2} - w_3\overrightarrow{v_3} \qquad (1)$$

Comme nous voulons assurer la continuité lors du déplacement d'un triangle, nous distribuons les pulsions en les pondérant avec les coordonnées barycentriques. Les changements de vitesse qui en résultent sont alors donnés par:

$$\Delta\overrightarrow{v_1} = \frac{w_1}{m_1} I^*\vec{n}\,;\ \Delta\overrightarrow{v_2} = \frac{w_2}{m_2}\overrightarrow{I^*n}\,;\ \Delta\overrightarrow{v_3} = \frac{w_3}{m_3} I^*\vec{n}\,;\ \Delta\overrightarrow{v_4} = \frac{1}{m_4} I^*\vec{n} \qquad (2)$$

Où m_i est la masse du sommet correspondant. En insérant l'équation (2) dans (1), nous obtenons l'impulsion I^* qui doit être appliquée:

$$I^*\vec{n} = \frac{\overrightarrow{v_{relN}}}{\frac{1}{m_4}+\frac{w_1^2}{m_1}+\frac{w_2^2}{m_2}+\frac{w_3^2}{m_3}} \qquad (3)$$

En outre, les relations respectives sont utilisées pour distribuer l'impulsion sur les sommets des primitives du maillage. Si les deux points en collision sont déjà plus proches qu'un seuil défini par l'utilisateur, par exemple, l'épaisseur du tissu, les deux primitives sont réduites à cette distance minimale pour résoudre la collision.

Pour éviter les problèmes visuels du glissement de tissu sur des objets statiques, nous allons modéliser des forces de frottement en utilisant la loi Coulombs. La vitesse relative des deux objets en contact est $\vec{v}_{rel}$. Sa composante dans la direction tangentielle est notée $\vec{v}_{relT}$. Soient $\overrightarrow{F_N}$ et $\overrightarrow{F_T}$ la force normale et la force tangentielle de frottement, alors les lois du frottement sont données par:

$$|\vec{v}_{relT}| \neq 0 \Longrightarrow \overrightarrow{F_T} = \mu\left|\overrightarrow{F_N}\right|\frac{\vec{v}_{relT}}{\vec{v}_{relT}} \qquad (4)$$

$$|\vec{v}_{relT}| = 0 \Longrightarrow \left|\overrightarrow{F_T}\right| < \mu\left|\overrightarrow{F_N}\right| \qquad (5)$$

Avec μ le coefficient de frottement. L'équation(4) modélise un contact glissant avec une force de frottement, l'équation (5) modélise un contact sans mouvement relatif. Ainsi, pour μ = 0, il existe un glissement sans frottement, alors que pour μ =∞ il n'y a pas de glissement. Pour adapter ces lois macroscopiques à la collision entre primitives, des simplifications concernant les forces normales et tangentielles sont nécessaires.

8. Algorithme de gestion de collision

Après avoir expliqué l'approche proposée, nous allons présenter l'algorithme de simulation qui englobe les différentes étapes de notre travail.

```
Algorithme simulation
Variable
Niv_res = 2 ;
```

```
{
    Saisie données ( ) ;
    Construction maillage (A) ;
    Construction  maillage (B) ;
    Construction  système masse-ressort ( ) ;
    Construction  hiérarchie (A,B);
While !Fin simulation do
Traverse (HA, HB);
End while
}
```

L'algorithme ci-dessus représente le processus de simulation générale, il est composé de différentes procédures et fonctions qui seront détaillées dans le prochain chapitre.

La fonction Traverse () représente le noyau de notre système, elle englobe le parcours de la hiérarchie et le test de collision à chaque pas du parcours ainsi que l'application du processus de multi résolution, la détection exacte et le traitement de collisions. Cette fonction est représentée par le pseudo-code suivant :

```
Traverse (HA,HB)
{
Parcours (HA, HB)
If (A, B ne s'intersectent pas) then
End if
Else
      If (a et b sont des feuilles) then
Return (FA, FB)
      While niv_res≠0 do
Multi_ resolution (FA, FB)
Construction hiérarchie (HFA, HFB)
niv_res - -
                Traverse (HFA, HFB)
      End while
Détectionexacte ()
Traitement collision ()
      Else
                For all children A[i], B[ j ]
                      Traverse (A[ i ], B[ j ])
                End for
      End if
End if
}
```

9. Conclusion

Durant ce chapitre, nous avons présenté l'approche générale de notre travail qui consiste à accélérer le processus de détection de collisions en utilisant la méthode des volumes englobants et en lui intégrant la notion de multi résolution. Nous avons commencé par donner une vue générale sur le système de simulation de tissu, en ciblant le module de gestion de

collision. Ceci en détaillant les différentes phases du processus et les différents composants du système. A commencer par la construction, la mise à jour ainsi que le parcours et le test de la hiérarchie. Ensuite nous avons détaillé le procédé qui nous a permis d'associer la multi résolution, en adaptant la technique de Hutchinson sur les maillages triangulés. La détection de collisions exactes entre deux primitives a été détaillée. Nous avons terminé ce chapitre par la présentation de la méthode de traitement de collisions choisie. Dans le chapitre suivant, nous allons donner les détails concernant l'implémentation de notre système et discuter les résultats obtenus.

CHAPITRE V

IMPLÉMENTATION ET RÉSULTATS

1. Introduction

Ce chapitre a pour but l'expérimentation et la validation du modèle proposé dans le chapitre précédent, à travers la réalisation d'une simulation. Pour ce faire, nous utilisons un système masse-ressort représentant notre tissu ainsi qu'une balle sphérique. À travers ce modèle, nous essayons d'animer cette scène en mettant la balle en contact avec le tissu afin de gérer les mouvements générés par la collision provoquée.

Le modèle a été construit en trois phases : la première phase consiste en la modélisation de la scène qui inclus la simulation du tissu par le système masse-ressort et la simulation du mouvement de la balle, la deuxième phase englobe la construction de la hiérarchie des sphères englobantes et la définition des opérations qui l'accompagnent alors que la troisième phase s'intéresse à la définition des différents aspects de la multi-résolution et présentons le module de gestion des collisions.

2. Environnement de travail

Notre modèle a été simulé par une application implémentée dans l'environnement Visual Studio 2008 avec le langage de programmation C++. Ce choix est justifié par les différentes qualités que procure cet environnement comme la rapidité et la portabilité. De plus, l'environnement intègre comme support de la bibliothèque graphique OpenGL.

Il existe plusieurs logiciels qui modélisent et simulent le tissu comme Blender, Maya, 3DSMax et des bibliothèques qui implémentent la gestion des collisions dans tous leurs états. Cependant, nous avons choisi de travailler avec des outils basiques pour éviter l'influence de ces bibliothèques sur notre travail.

Notre programme a été testé sur un ordinateur basé sur le processeur Intel(R) Core (TM) i5 avec 4 Giga-octets de RAM et une carte graphique Intel possédant une mémoire dédiée de 1 Giga-octet.

3. Architecture globale de l'implémentation

Le schéma ci-dessous montre les relations entre les différents techniques faisant partie du système global implémenté afin d'accomplir le but recherché dans ce travail, ce but étant de gérer les collisions survenues lors de la simulation du tissu en interaction avec la boule en mouvement.

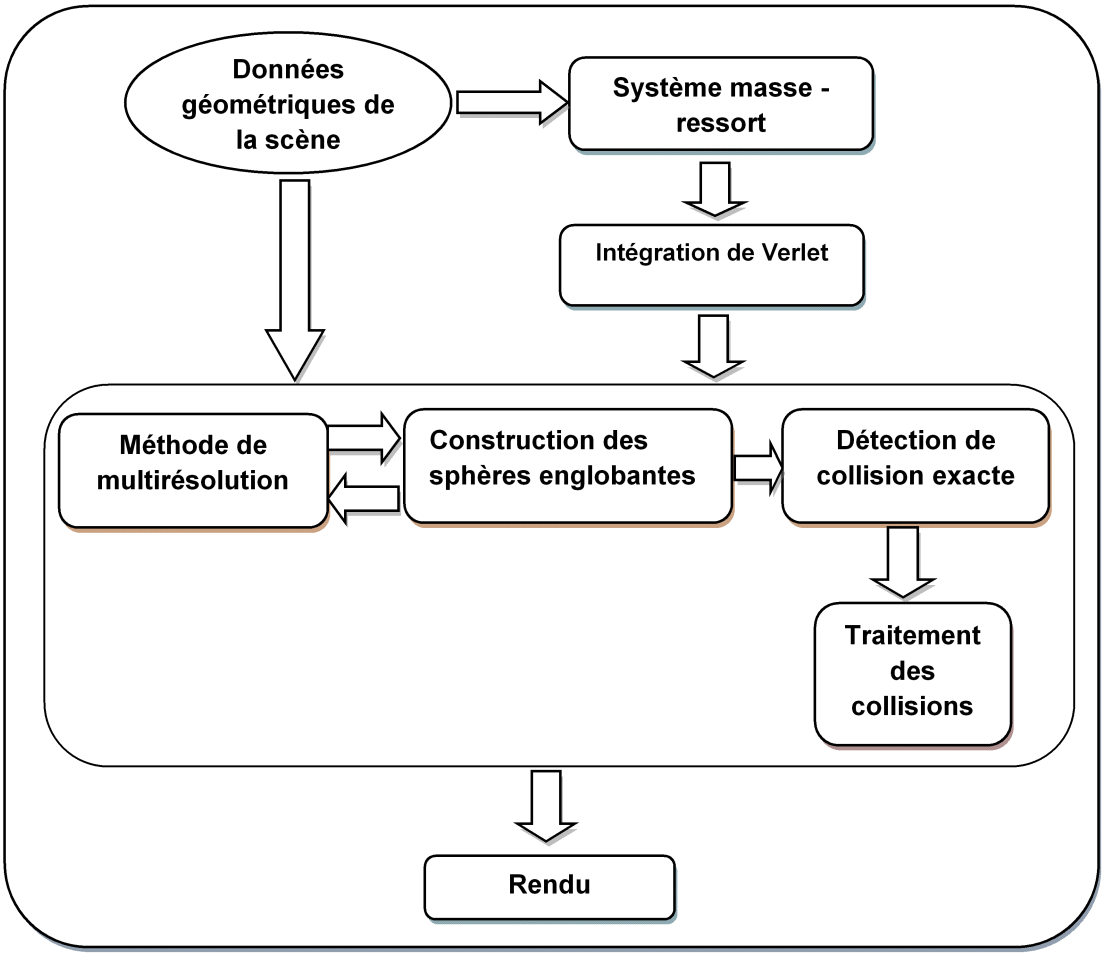


Figure V.1 Schéma général du système implémenté.

Ce schéma retrace les étapes essentielles de la simulation que nous sommes proposées de réaliser:

- La simulation du tissu : elle consiste à modéliser ce dernier avec le système masse-ressort en utilisant l'intégration de Verlet pour le calcul des nouvelles positions. A ce niveau, nous avons préparé la représentation grossière de notre tissu. Celle-ci représente le paramètre d'entrée de l'algorithme de multi-résolution ainsi que celui de construction de la hiérarchie des sphères englobant. Les procédures utilisées à ce niveau sont : timeStep(), addForce (), windForce (), satisfyConstraint ().

- Au deuxième niveau de notre simulation, intervient l'algorithme de construction des sphères englobantes : cet algorithme prend les particules du système masse-ressort avec une représentation grossière et construit la sphère minimale englobant ces particules en utilisant la fonction MinSphere(), RecurseMin(). Ensuite, la hiérarchie est construite avec les procédures BuidBSH(), RecurseBuild().
- La multi-résolution intervient au troisième niveau de notre simulation, et ce en utilisant la procédure traverse(). De plus, le test des collisions et leur traitement avec la fonction doesItCollide() et la procédure ballCollision().
- A ce niveau, interviennent le rendu et l'affichage qui utilisent les procédures drawShaded(), drawTriangle() et display().

Les fonctions et procédures citées ci-dessus sont détaillées plus.

4. Classes et structures de données utilisées

Dans ce système, nous avons utilisé différentes classes et structures de données dont nous citons celles considérées les plus comme étant essentielles sont :

- class Vec3 : c'est une classe d'un vecteur minimal de trois réels, avec les opérateurs mathématiques surchargés.
- class Particule : cette classe concerne les particules du système masse-ressort, ses attributs sont :
  - *mass* : qui représente la masse de la particule, dans notre modèle elle est initialisée à 1.
  - *pos* : représente la position courante dans l'espace 3D.
  - *old_pos :* représente la position de la particule dans le pas de temps précédent et qui est utilisée dans le schéma d'intégration de Verlet.
  - *Acceleration* : un vecteur représentant l'accélération courante de la particule.
  - *accumulated_normal* : la normale non normalisée utilisée dans OpenGL soft shading.
  - *State* : représente l'état de la particule, 1 si elle est active 0 sinon.
  - *particleIndex* : représente l'index de la particule dans le vecteur de particule selon l'ordre de sa création.

  Les méthodes de cette classe sont : resetAcceleration(), offsetPos(), makeUnmovable(), void addToNormal(), getNormal(),resetNormal();
- class Constraint : cette classe est construite pour pouvoir établir les connexions entre les particules du système masse-ressort. Elle a comme attributs deux particules p1 et p2 ainsi

que rest_distance qui est la distance entre deux particules dans un état de repos. Cette classe a comme méthode la procédure satisfyConstraint() qui permet de satisfaire les contraintes entre deux particules.

- class Cloth : cette classe contient comme attributs les différentes structures de données manipulées lors de la simulation du tissu, comme les vecteurs de particules et de contraintes 'particleActiveState' et 'Constraints', ainsi que la matrice d'activation des particules 'MatResol'. Les méthodes utilisées sont calcTriangleNormal(), addWindForcesForTriangle(), drawTriangle(), drawShaded(),timeStep(),addForce() et windForce().
- class BSH : c'est la classe à travers laquelle nous construisons la hiérarchie des sphères englobantes. Elle contient comme méthodes les procédures BuidBSH(), RecurseBuild(), SearchBSH(), RecurseSearch(),traverse(), connectParticles() et doesItCollide.
- class Sphère : c'est la classe à travers laquelle nous construisons les sphères minimales dont nous avons besoin pour la construction de la hiérarchie des sphères englobantes. Elle contient les procédures suivantes : MinSphere(), RecurseMin().

5. Algorithmes utilisés

Les fonctions et procédures citées ci-dessus seront détaillées, nous allons commencer par les procédures qui mettent en œuvre le système masse-ressort.

5.1 Mise en œuvre du système masse-ressort

Le système masse-ressort est construit à partir des particules contenues dans le vecteur 'particleActiveState' en établissant les connexions entres elles, ceci étant fait grâce à la fonction makeConstraint(). Une fois les connexions établies, nous appliquons les forces du vent et de la gravité sur chaque triangle issu de la fonction de connexion des particules, à travers les procédures calcTriangleNormal(), addWindForcesForTriangle(), drawTriangle(), addForce(), windForce(). Après avoir affecté les différentes forces aux triangles de notre maillage, nous procédons à son dessin en utilisant la procédure drawShaded(), la mise à jour des positions et des accélérations est réalisée par la procédure timeStep().

Le fragment de code suivant représente la construction des ressorts de structure et de cisaillement.

Connexion des voisins immédiats avec des ressorts.

```
for(int x=0; x<num_particles_width; x+=4)
```

```
  {
    for(int y=0; y<num_particles_height; y+=4)
      {
if(x<num_particles_width-4) makeConstraint(getParticle2(x,y),getParticle2(x+4,y));
if(y<num_particles_height-4)
       makeConstraint(getParticle2(x,y),getParticle2(x,y+4));
if(x<num_particles_width-4&&y<num_particles_height-4)
       makeConstraint(getParticle2(x,y),getParticle2(x+4,y+4));
if(x<num_particles_width-4&&y<num_particles_height-4)
        makeConstraint(getParticle2(x+4,y),getParticle2(x,y+4));
      }
  }
```

Ensuite, nous construisons les ressorts de flexion comme suis :

Connexion des voisins seconds avec des ressorts.

```
for(int x=0; x<num_particles_width; x+=4)
{
  for(int y=0; y<num_particles_height; y+=4)
  {
      if(x<num_particles_width-4)
      makeConstraint(getParticle2(x,y),getParticle2(x+4,y));

      if(y<num_particles_height-4)
      makeConstraint(getParticle2(x,y),getParticle2(x,y+4));

      if(x<num_particles_width-4&&y<num_particles_height-4)
      makeConstraint(getParticle2(x,y),getParticle2(x+4,y+4));

      if(x<num_particles_width-4&&y<num_particles_height-4)
      makeConstraint(getParticle2(x+4,y),getParticle2(x,y+4));
   }
}
```

Les deux fonctions précédentes font appel à la procédure makeConstraint.

La procédure makeConstraint.

```
Void  makeConstraint(Particle *p1, Particle *p2)
{
constraints.push_back(Constraint(p1,p2));
}
```

Après avoir construit le système masse-ressort, nous appliquons les forces du vent et de gravité.

Ajouter la force de gravité à chaque particule.

```
void addForce(const Vec3 direction)
{
std::vector<Particle>::iterator particle;
for(particle = particles.begin(); particle != particles.end(); particle++)
  {
      (*particle).addForce(direction);
  }
}
```

Ajouter la force du vent à chaque particule.

```
void windForce(const Vec3 direction)
{
 for(int x = 0; x<num_particles_width-4; x+=4)
  {
   for(int y=0; y<num_particles_height-4; y+=4)
        {
         addWindForcesForTriangle(getParticle2(x+4,y),getParticle2(x,y),
         getParticle2(x,y+4),direction);
         addWindForcesForTriangle(getParticle2(x+4,y+4),getParticle2(x+4,y),
         getParticle2(x,y+4),direction);
       }
   }
}
```

Les deux procédures précédentes utilisent la procédure addWindForcesForTriangle().

La procédure addWindForcesForTriangle()

```
Void addWindForcesForTriangle(Particle *p1,Particle *p2,Particle *p3, const
Vec3 direction)

{

   Vec3 normal = calcTriangleNormal(p1,p2,p3);

   Vec3 d = normal.normalized();

   Vec3 force = normal*(d.dot(direction));

    p1->addForce(force);

    p2->addForce(force);

    p3->addForce(force);

}
```

5.2 Construction des Sphères Minimales

Cet algorithme calcule le plus petit disque englobant d'un nombre fini de points dans le plan en temps linéaire. Il est représenté par les procédures suivantes :

Fonction construisant la Min Sphère en appelant récursivement RecurseMin

```
Sphere MinSphere( Point P[], unsigned int p )
{
      Point **L = new Point*[p];
      for(unsigned int i = 0; i < p; i++)
            L[i] = &P[i];
      Sphere MB = RecurseMin(L, p);
      delete[] L;
      return MB;
}
```

Fonction RecurseMin

```
Sphere RecurseMin( Point *P[], unsigned int p, unsigned int b )
{
      Sphere MB;
      switch(b)
      {
      case 0:
```

```
                MB = Sphere();
                break;
          case 1:
                MB = Sphere(*P[-1]);
                break;
          case 2:
                MB = Sphere(*P[-1], *P[-2]);
                break;
          case 3:
                MB = Sphere(*P[-1], *P[-2], *P[-3]);
                break;
          case 4:
                MB = Sphere(*P[-1], *P[-2], *P[-3], *P[-4]);
                return MB;
          }

          for(unsigned int i = 0; i < p; i++)
                if(MB.d2(*P[i]) > 0)     // Signed square distance to
    sphere
                {
                      for(unsigned int j = i; j > 0; j--)
                      {
                            Point *T = P[j];
                            P[j] = P[j - 1];
                            P[j - 1] = T;
                      }

                      MB = RecurseMin(P + 1, i, b + 1);
                }

                return MB;
    }
```

5.3 Construction de la hiérarchie

Cet algorithme représente la construction d'un arbre où chaque nœud constitue une sphère qui englobe un nombre de faces et possède un fils gauche et un fils droit. Le niveau de la hiérarchie est défini par le programmeur. L'algorithme est composé de plusieurs procédures telles que BuildBSH(), RecurseBuild(), SearchBSH() et RecurseSearch(), les plus importantes sont BuildBSH(), RecurseBuild().

Fonction construisant la BSH en appelant récursivement RecurseBuild().

```
   void BSH::BuidBSH( objLoader *model, int level, Cloth cloth )
   {
      this->model=model;
      this->level=level;

      Sphere BS;
      Point *P = new Point[cloth.particleActiveState.size()];

      int k,l=0;
```

```
for (k=0;k<cloth.particleActiveState.size();k=k+276)
  {
      for (;l<69+k;l=l+4)
      {     obj_vector* v=model->vertexList[l];
            P[l].x=v->e[0];
          P[l].y=v->e[1];
          P[l].z=v->e[2];
      }
      l=l-3;
   l=l+207;
 }

BS = MinSphere(P, model->vertexCount);

root->BS->center=BS.center;
root->BS->radius=BS.radius;
root->pere=NULL;
FaceList *faces=new FaceList;
for(int i=0; i<model->faceCount; i++)
{
      obj_face *o = model->faceList[i];

      obj_vector* v[3];

      for (int j=0;j<3;j++)
      {
        v[j]=model->vertexList[o->vertex_index[j]];
      }

      vector3 sum(0,0,0);

      for(int j=0; j<3; j++)
      {
            sum.x+=v[j]->e[0];
            sum.y+=v[j]->e[1];
            sum.z+=v[j]->e[2];
      }

      sum.x/=3;
      sum.y/=3;
      sum.z/=3;

      Face tri;
      tri.center=sum;
      tri.index=i;

      faces->push_back(tri);
}

if (level<=1)
{
      faces->clear();
      delete faces;
      return;
```

```
            }

            RecurseBuild(root,faces,2);

            faces->clear();
            delete faces;
  }
```

La fonction précédente fait appel à RecurseBuild()

Fonction RecurseBuild()

```
      void BSH::RecurseBuild( BSN *current, FaceList *faces, int l )
      {

            float minX, maxX, minY, maxY, minZ, maxZ;
            float sum[3];

            sum[0]=sum[1]=sum[2]=0;

            minX=minY=minZ=1e20;
            maxX=maxY=maxZ=-minX;

            for (int i=0;i<faces->size();i++)
            {
                  Face tri=faces->at(i);

                  sum[0]+=tri.center.x;
                  sum[1]+=tri.center.y;
                  sum[2]+=tri.center.z;

                  if (tri.center.x<minX)
                  minX=tri.center.x;

                  if (tri.center.x>maxX)
                  maxX=tri.center.x;

                  if (tri.center.y<minY)
                  minY=tri.center.y;

                  if (tri.center.y>maxY)
                  maxY=tri.center.y;

                  if (tri.center.z<minZ)
                  minZ=tri.center.z;

                  if (tri.center.z>maxZ)
                  maxZ=tri.center.z;
            }

            float span[3];
```

```
span[0]=maxX-minX;
span[1]=maxY-minY;
span[2]=maxZ-minZ;

int axis;

if (span[0]>span[1])
{
      if (span[0]>span[2])
      {
            axis=0;
      }else{
            axis=2;
      }
}else{
      if (span[1]>span[2])
      {
            axis=1;
      }else{
            axis=2;
      }
}
sum[axis]/=faces->size();
FaceList *leftFace=new FaceList;
FaceList *rightFace=new FaceList;
for (int i=0;i<faces->size();i++)
{
      Face tri=faces->at(i);

      bool left=false;

      switch (axis)
      {
      case 0:
            if (sum[axis]<tri.center.x)
            {
                  left=true;
            }
            break;

      case 1:
            if (sum[axis]<tri.center.y)
            {
                  left=true;
            }
            break;

      case 2:
            if (sum[axis]<tri.center.z)
            {
                  left=true;
            }
            break;
      }
      if (left)
      {
            leftFace->push_back(tri);
```

```
		}else{
			rightFace->push_back(tri);
		}
	}

		//La Construction des nœuds gauches//

	Sphere lBS;
	Point *lP = new Point[leftFace->size()*3];

	for (int i=0;i<leftFace->size();i++)
	{
		obj_face *o = model->faceList[leftFace->at(i).index];

		obj_vector* v[3];

		for (int j=0;j<3;j++)
		{
			v[j]=model->vertexList[o->vertex_index[j]];
		}

		for (int j=0;j<3;j++)
		{
			lP[i*3+j].x=v[j]->e[0];
			lP[i*3+j].y=v[j]->e[1];
			lP[i*3+j].z=v[j]->e[2];
		}
	}

	lBS = MinSphere(lP, leftFace->size()*3);

	current->left=new BSN;
	current->left->BS=new Sphere();
	current->left->pere=current;
	current->left->BS->center=lBS.center;
	current->left->BS->radius=lBS.radius;

	if (l==level)
	{
		current->left->left=NULL;
		current->left->right=NULL;
		current->left->mesh=new TriList;

		for (int i=0;i<leftFace->size();i++)
		{
			int ind=leftFace->at(i).index;
			current->left->mesh->push_back(ind);
		}
	}else{
		RecurseBuild(current->left,leftFace,l+1);
	}

	leftFace->clear();

	delete [] lP;
```

```
        delete leftFace;


                ///construction des nœuds de la droite///

        Sphere rBS;
        Point *rP = new Point[rightFace->size()*3];

        for (int i=0;i<rightFace->size();i++)
        {
              obj_face *o = model->faceList[rightFace->at(i).index];

              obj_vector* v[3];

              for (int j=0;j<3;j++)
              {
                    v[j]=model->vertexList[o->vertex_index[j]];
              }

              for (int j=0;j<3;j++)
              {
                    rP[i*3+j].x=v[j]->e[0];
                    rP[i*3+j].y=v[j]->e[1];
                    rP[i*3+j].z=v[j]->e[2];
              }
        }
        rBS = MinSphere(rP, rightFace->size()*3);

        current->right=new BSN;
        current->right->BS=new Sphere();
        current->right->pere=current;
        current->right->BS->center=rBS.center;
        current->right->BS->radius=rBS.radius;

        if (l==level)
        {
              current->right->left=NULL;
              current->right->right=NULL;
              current->right->mesh=new TriList;

              for (int i=0;i<rightFace->size();i++){
                    int ind=rightFace->at(i).index;
                    current->right->mesh->push_back(ind);
              }
        }else{
              RecurseBuild(current->right,rightFace,l+1);
        }
        rightFace->clear();
        delete [] rP;
        delete rightFace;
      }
```

5.4 La multi résolution

L'intégration de la multi résolution se fait lors du parcours de la hiérarchie des sphères englobantes. Le parcours de l'arbre se fait en largeur parce que nous avons besoin de tester s'il existe une collision potentielle à chaque niveau de l'arbre. Chaque niveau contient deux nœuds, gauche et droit, et nous testons sur la collision par rapport aux deux nœuds si on ne trouve pas une collision potentielle avec un des nœuds on le supprime et on continue le parcours jusqu'à arriver aux nœuds feuilles, où nous appliquons notre subdivision qui consiste à activer les particules inactives situées dans le voisinage de chaque particule appartenant à la zone de collision potentielle.

Les procédures utilisées sont les suivantes :

La procédure Traverse ()

```
void traverse(Cloth cloth, BSN *current,const Vec3 center,const float
radius,
int Reslevel,float stepsize)
{
 queue <BSN*> file;
 int i=-1; int line,col,col1;
 file.push(current);
 if(!doesItCollide(current->BS, center, radius))
 return;
 while(!file.empty())
      {
       BSN *node=file.front();
       i++;
       file.pop();
       if(doesItCollide(node->BS, center, radius))
        {
         if(node->left==NULL&&node->right==NULL)
          {
           leaffaces->push_back(node->mesh);
           }
           else
           {
            if(node->left!=NULL)
            file.push(node->left);
            if(node->right!=NULL)
            file.push(node->right);
           }
         }
         else
          {
           if(i%2!=0) node->pere->left=NULL;
            else       node->pere->right=NULL;
        }
     }
      TriList *Refine=new TriList;
      TriList *Refine1=new TriList;
      for(int i=0;i<leaffaces->size();i++)
       {
         for(int j=0;j<leaffaces->at(i)->size(); j++)
           {

           Refine->push_back(cloth.Indexface[leaffaces->at(i)->at(j)]-
      >vertex_index[0]);
           Refine->push_back(cloth.Indexface[leaffaces->at(i)->at(j)]-
      >vertex_index[1]);
```

```
          Refine->push_back(cloth.Indexface[leaffaces->at(i)->at(j)]-
      >vertex_index[2]);
          }
        }
      if(!Refine->empty())
      {
        Refine1->push_back(Refine->at(0));
        int ind; vector<int>::iterator it;
        vector<int>::iterator it1;
        for(it1=Refine->begin();it1<Refine->end();it1++)
      {
        ind=*it1;
        for(it=Refine1->begin();it<Refine1->end()&&(*it)!=ind;it++);
        if(it>=Refine1->end())
        Refine1->push_back(ind);
      }

      //////////Activer le voisinage de la particule en collision//////
      int l=0;
      for(int i=0;i<Refine1->size();i++)
      {
        for(l=0;l<cloth.t->size()&& cloth.t->at(l).index!=Refine1-
      >at(i);l++);
          if(l<cloth.t->size())
            { line=col=col1=0;
              int j= cloth.t->at(l).x;
              int k= cloth.t->at(l).y;

         if(k==0&&j!=0&&j!=(cloth.num_particles_height* (Reslevel+1)-
      Reslevel)-1)
           { line=5; col=9; col1=9;
            for(int m=0;m<k+5;m++)
            for(int n=j-4;n<j+5;n++)


cloth.RefinedParticles.push_back(cloth.particleActiveState.at(cloth.MatRe
sol[m][n]));

            }
          else
          if(k==(cloth.num_particles_height* (Reslevel+1)-Reslevel)-
      1&&j!=0
             &&j!=(cloth.num_particles_height* (Reslevel+1)-Reslevel)-1)
           { line=5; col=9; col1=9;
           for(int m=k-4;m<=k;m++)
               for(int n=j-4;n<j+5;n++)


cloth.RefinedParticles.push_back(cloth.particleActiveState.at(cloth.MatRe
sol[m][n]));

           }
           else
```

```
           if(j==0&&k!=0&&k!=(cloth.num_particles_height* (Reslevel+1)-
      Reslevel)-1)
           {
             line=9; col=5; col1=5;
             for(int m=k-4;m<k+5;m++)
            for(int n=0;n<j+5;n++)
      cloth.RefinedParticles.push_back(cloth.particleActiveState.at(cloth
      .MatResol[m][n]));
          }
          else
          if(j==(cloth.num_particles_height* (Reslevel+1)-Reslevel)-1&&
              k!=0&&k!=(cloth.num_particles_height* (Reslevel+1)-
      Reslevel)-1)
           {
            line=9; col=5; col1=5;
            for(int m=k-4;m<k+5;m++)
            for(int n=j-4;n<=j;n++)

cloth.RefinedParticles.push_back(cloth.particleActiveState.at(cloth.MatRe
sol[m][n]));
           }
           else
           if(k==0&&j==0)
           {
           line=5; col=5; col1=5;
           for(int m=0;m<k+5;m++)
           for(int n=0;n<j+5;n++)
      cloth.RefinedParticles.push_back(cloth.particleActiveState.at(cloth
      .MatResol[m][n]));
           }
          else
          if((k==(cloth.num_particles_height* (Reslevel+1)-Reslevel)-1)&&
          (j==(cloth.num_particles_height* (Reslevel+1)-Reslevel)-1))
           {
           line=5; col=5; col1=5;
           for(int m=k-4;m<=k;m++)
           for(int n=j-4;n<=j;n++)
      cloth.RefinedParticles.push_back(cloth.particleActiveState.at(cloth
      .MatResol[m][n]));
           }
           else
           if(k==0&&j==(cloth.num_particles_height* (Reslevel+1)-
      Reslevel)-1)
           {
           line=5; col=5; col1=5;
           for(int m=0;m<k+5;m++)
           for(int n=j-4;n<=j;n++)
      cloth.RefinedParticles.push_back(cloth.particleActiveState.at(cloth
      .MatResol[m][n]));
           }
           else
           if(k==(cloth.num_particles_height* (Reslevel+1)-Reslevel)-
      1&&j==0)
           {
```

```
            line=5; col=5; col1=5;
            for(int m=k-4;m<=k;m++)
            for(int n=0;n<j+5;n++)

cloth.RefinedParticles.push_back(cloth.particleActiveState.at(cloth.MatRe
sol[m][n]));
      }
            else
            {
            line=9; col=9; col1=9;
            for(int m=k-4;m<k+5;m++)
            for(int n=j-4;n<j+5;n++)
      cloth.RefinedParticles.push_back(cloth.particleActiveState.at(cloth
      .MatResol[m][n]));
             }
}
```

5.5 Gestion des collisions

Ce module comporte la détection basique de la collision qui constitue le test si la particule est à l'intérieur de la sphère. Ceci se fait par la comparaison du vecteur de distance entre la particule et le centre de la sphère et le rayon de cette dernière

Fonction construisant la Min Sphère en appelant récursivement RecurseMin

```
void ballCollision1(const Vec3 center,const float radius )
  {
   std::vector<Particle>::iterator particle;
   for(int i=0;i<RefinedParticles.size();i++)
    {
      for(particle    =    particleActiveState.begin();    particle    !=
      particleActiveState.end()

      &&(RefinedParticles.at(i).particleIndex!=(*particle).particleIndex);
            particle++);
         if(particle<particleActiveState.end())
           {
              Vec3 v = (*particle).getPos()-center;
              float l = v.length();
              if ( v.length() < radius) // if the particle is inside the
      ball
                  {
                    (*particle).offsetPos(v.normalized()*(radius-l));
                  }
                  }

                }
      }
```

6. Mise en œuvre de la simulation

Notre système consiste à simuler un tissu en interaction avec une boule en mouvement. La simulation débute par la modélisation du tissu en utilisant le système masse-ressort, et ce par l'usage de l'intégration de Verlet pour le calcul des nouvelles positions. A ce niveau, nous avons préparé la représentation grossière de notre tissu. Celle-ci représente le paramètre d'entrée de l'algorithme de la multi-résolution ainsi que celui de la construction de la hiérarchie des sphères englobantes.

Au deuxième niveau de notre simulation intervient l'algorithme de la construction des sphères englobantes : cet algorithme prend les particules du système masse ressort dans la représentation grossière et construit la sphère minimale englobant ces particules. Ensuite, chaque sphère ainsi construite est subdivisée en deux ensembles de particules où chaque ensemble est englobé par une sphère minimale. Les sphères résultantes permettent de construire notre hiérarchie qui est représentée par un arbre binaire. Chaque nœud de cet arbre binaire constitue une sphère qui englobe un nombre de faces. Le niveau de la hiérarchie est défini par le programmeur. Une fois la hiérarchie construite, elle est parcourue en largeur parce que nous avons besoin de tester s'il existe une collision potentielle à chaque niveau de l'arbre. Chaque niveau contient deux nœuds, gauche et droit, nous testons sur la collision par rapport aux deux nœuds et si on ne trouve pas une collision potentielle avec un des nœuds on le supprime et on continue le parcours jusqu'à arriver aux nœuds feuilles, où nous nôtres méthode de subdivision qui consiste à activer les particules inactives situées dans le voisinage de chaque particule appartenant à la zone de collision potentielle. Nous testons et traitons ensuite les collisions exactes. Ce processus se fait dans un pas de simulation et se répète pendant la boucle de simulation.

6.1 Condition initiale de l'application

Pour être déroulée, notre simulation a besoin de quelques constantes physiques, ces constantes sont :

- le coefficient de l'élasticité égale à 0.01 ;
- le pas de temps qui est égal à 0.5 ;
- le nombre de particules en largeur et en hauteur du niveau grossier et qui est égal à 18 x18 ;
- La largeur et la hauteur du tissu égale à 10x10 ;
- Le niveau de la hiérarchie des sphères englobant qui est 15 ;
- Le rayon et le centre de la sphère en mouvement.

6.2 Résultats obtenus

Les résultats suivants sont obtenus lors du déroulement de la simulation. Ils représentent le modèle dans différents cas de figure, et ce en variant le nombre de particules entre représentation fine et grossière. Et en affichant le tissu avec rendu et sans rendu ainsi que dans le cas où il y a collision et dans le cas où il n'y a pas collision. Et enfin en affichant les sphères englobantes et sans leur affichage.

La figure suivante représente le tissu sous une représentation grossière, avec les différents cas de figure.

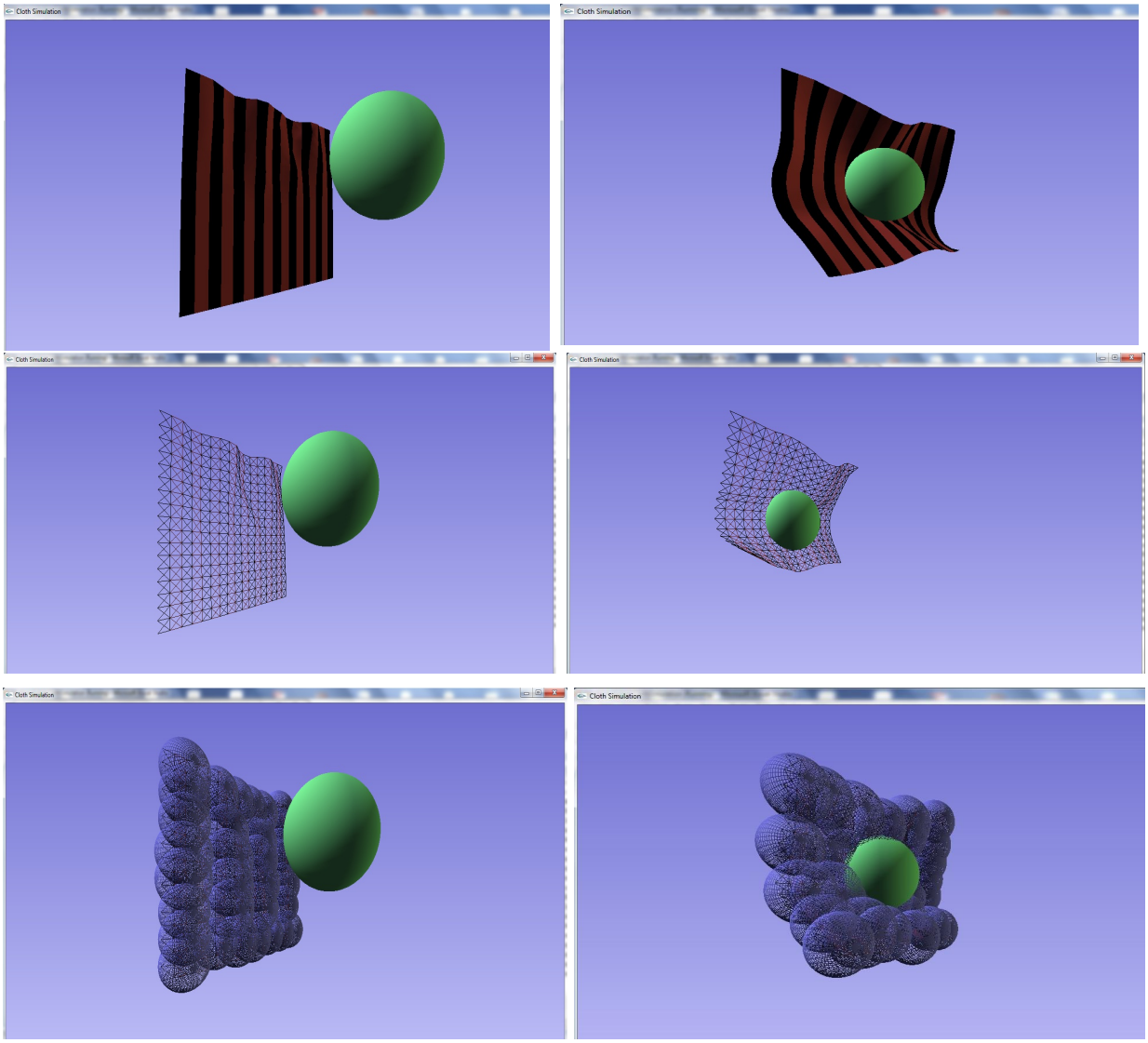


Figure V.2 Représentation du tissu avec un niveau de détail grossier. Différents cas de figure.

La figure suivante représente le tissu sous une représentation détaillée, avec les différents cas de figure.

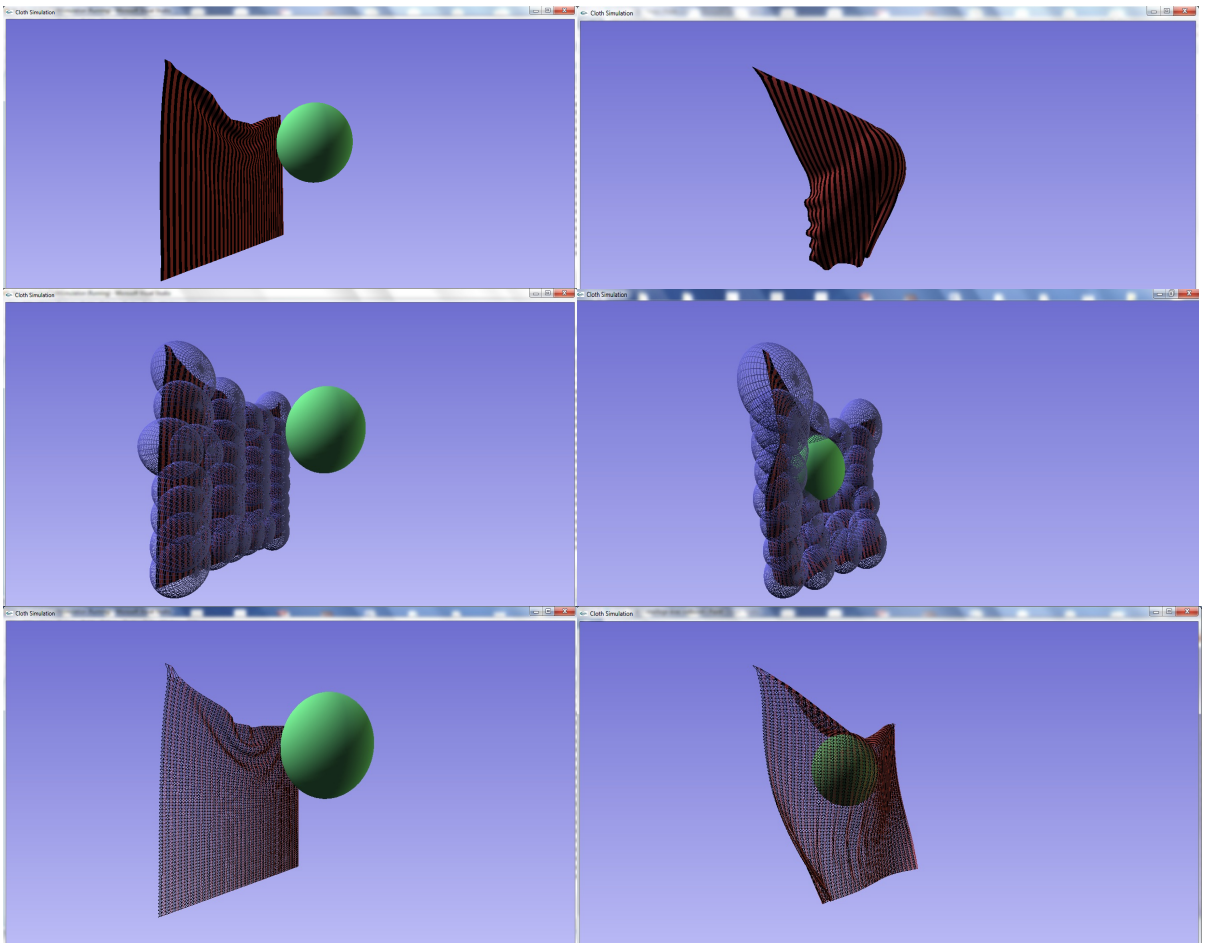


Figure V.3 Représentation du tissu avec un niveau de détail raffiné. Différents cas de figure.

6.3 Analyse des résultats

Durant le déroulement de la simulation nous avons conclu que la hiérarchie des sphères englobantes nous procure un compromis entre qualité et performance parce que d'autant que nous avons besoin d'un nombre important de sphères pour approximer les déformations du tissu et donc un niveau important pour la profondeur de l'arbre, la performance n'était pas altérée et ceci à cause des tests faciles de la collision point/sphère. Ainsi, nous avons gagné en qualité tout en gardant stable la performance.

La multi résolution a permis, en l'associant à la hiérarchie, de minimiser le parcours de 69x69 particules pour le test de collision. Cette méthode a permis de cerner la collision en ne raffinant que la région délimitée par le voisinage de la particule concernée par la collision potentielle. Nous avons aussi conclu que, plus le niveau de l'arbre est grand, plus la détection

est rapide jusqu'à une certaine limite. Ceci est dû au fait que la région de collision potentielle est réduite quand on a un arbre plus profond.

7. Conclusion

La réalisation de notre outil se résume globalement dans la construction d'un système masse ressort ayant la possibilité de réaction en appliquant les forces de gravité et du vent. Ce système représente notre tissu qui entrera en interaction avec une boule en mouvement.

L'objectif du travail consiste à accélérer la détection des collisions en intégrant la multi résolution dans une hiérarchie de sphères englobantes représentant le tissu. L'accélération permet de réduire le temps de calcul en gardant stable la qualité de la simulation. Les résultats obtenus peuvent être jugés satisfaisants comparant au même système simulé avec le même nombre de particules sans utilisation des méthodes accélératrices.

Dans ce chapitre nous avons décrit les différents algorithmes utilisés lors de la réalisation du système ainsi que les différentes classes et structures de données. Nous avons ensuite décrit les résultats obtenus et les analyses effectuées.

Conclusion générale et perspectives

Deux décennies de recherche dans la simulation physique ont transformé le rêve des animateurs d'avoir des simulations temps réel en une réalité.

Toutefois, la simulation des vêtements et du tissu en général reste perturbante puisqu'on n'arrive pas encore à reproduire une animation réaliste, interactive et en temps réel du tissu, du fait que pour gagner en qualité, la simulation perd en termes de temps de calcul.

Pour la simulation du tissu, nous utilisons les systèmes masse-ressort, qui permettent de simuler le phénomène de l'élasticité. Pour réaliser ce système, deux points doivent être principalement résolus:

1. Pour déterminer la position des particules du tissu à chaque pas de temps, un grand système d'équations différentielles représentant l'ensemble du maillage du, doit être résolu.
2. Pour garantir le comportement correct du mouvement, les collisions entre le tissu et l'environnement externe et les auto-collisions doivent être détectées et corrigées.

Ce processus, appelé gestion de collisions est très lent et représente un goulot d'étranglement pour chaque simulation incluant l'animation des tissus. Ceci est dû au parcours par l'algorithme chargé du traitement des collisions de chaque primitive du maillage représentant le tissu et cela à chaque pas de temps. Pour cela, différentes méthodes accélératrices sont employées pour essayer de résoudre ce problème de lenteur et trouver un compromis entre qualité et vitesse de calcul.

Notre travail s'inscrit dans le cadre de la simulation comportementale. Son principal objectif est de proposer une solution pour la simulation des tissus en temps réel. Plus précisément, le développement d'un ensemble de méthodes pour résoudre les différents problèmes que pose l'animation temps réel des tissus, ceci dans le but de reproduire de la façon la plus fidèle et la plus précise possible le comportement des tissus ainsi que la gestion des collisions qu'ils génèrent, et ceci en tirant profit des

techniques des niveaux de détails pour leurs propriétés d'accélération. L'objectif principal est de trouver un compromis entre la rapidité de calcul et la précision de la détection.

Le modèle proposé gère les collisions dans le cadre de la simulation de tissu et ce, en utilisant une méthode d'optimisation basée sur les volumes englobants et la technique de la multi résolution pour accélérer le processus de détection de collisions.

Nous avons commencé notre travail par choisir parmi toutes les méthodes accélératrices existantes celle qui convient le mieux dans le domaine de la simulation des objets hautement déformables tels que le tissu. Notre choix s'est porté sur les volumes englobants, qui sont les plus simples à appliquer parmi toutes les méthodes. Et parmi les volumes qui existent, nous avons utilisé les sphères qui réduisent le temps de calcul ainsi que l'espace mémoire vu que nous n'aurons besoin de calculer que la distance entre les sommets du maillage et les centres des sphères.

L'utilisation d'une méthode accélératrice dans notre travail visait à simplifier les tests de collision du tissu avec l'objet externe, qui sont extrêmement longs vu que nous considérant le tissu comme un maillage triangulaire. Cette méthode accélératrice a été combinée avec la méthode de la multi résolution dans le but d'améliorer la qualité de la simulation, en présentant les zones les plus susceptibles à entrer en collision avec une résolution supérieure (maillage détaillé), tandis que les zones qui ne seront pas sujettes à des collisions avec des résolutions inférieures (maillage grossier).

Ces deux méthodes principales ont été combinées pour réaliser notre système qui vise particulièrement à gérer les collisions d'une balle entrant en collision avec un bout de tissu.

Notre méthode a permis de réaliser une qualité convenable de la simulation, en utilisant un nombre important de sphères englobantes pour approximer les déformations du tissu. Ceci n'a pas altéré la performance de l'application, vu que les tests de collision point/sphère sont faciles à réaliser. D'autant plus que l'utilisation de la multi résolution a permis de minimiser le parcours du test de collision. cœur

Notre travail pourrait être amélioré, en utilisant la détection de collision continue qui vise à améliorer la qualité de la simulation. Ainsi que le parallélisme en utilisant une architecture CPU multi-cœur, de plus que les fonctionnalités des GPUs. L'utilisation des maillages non structurés améliorera la déformation des tissus, par la suite la qualité de la simulation. La prise en charge des froissements et des plis sera bénéfique en étendant la méthode sur la gestion des collisions entre les vêtements et le corps.

Notre méthode a été appliquée sur un système simple qui avait comme but d'étudier la gestion des collisions entre le tissu et un objet rigide en mouvement. Par conséquent, notre méthode peut être étendue à des applications plus complexes. Comme la gestion des collisions entre les vêtements et le corps humain et la simulation des chirurgies en gérant les collisions entre les outils chirurgicaux et les organes humains.

Références Bibliographiques

[1] K. Sundaraj,"Détection et traitement de collision en simulation dynamique," mémoire DEA, Institut National Polytechnique de Grenoble, 23 juin 2000.

[2] M. Hatab, "Vers une approche unifiée de détection de collision pour objets de natures diverses, " thèse de doctorat, Université d'Evry –val d'Essonne,15 Décembre 2006.

[3] P. Meseure, "Modélisation de corps déformables pour la simulation d'actes chirurgicaux," thèse de doctorat, Université des sciences et technologies de Lille, 1997.

[4] M. Nesme, "Détection en temps réel d'auto-collision sur des objets hautement déformables," mémoire de magistère, UFR IMA –Université Joseph Fourier, 2003.

[5] P. Meseure, A. Kheddar, F. Faure, "Détection des collisions et calcul de la réponse," Action Spécifique CNRS N°90, IRCOM/SIC Université de Poitiers, LSC Université d'Evry_Val d'Essonne, GRAVIR _ INPG Université Joseph Fourier, Grenoble, 2003.

[6] P. Meseure, "Animation basée sur la physique pour les environnements interactifs temps-réel," Habilitation à diriger les recherches, Université des sciences et technologies de Lille ,2002.

[7] S. Benameur, N.E. Djedi, "Intégration de la multi-résolution dans un système masse-ressort : application à l'animation de tissu," TAIMA'05, PP495-500, Tunisie 2005.

[8] L. Joussemet, "Approche évolutionniste pour la détection des collisions au sein d'environnements virtuels denses," Thèse de Doctorat, Sciences et Techniques du Languedoc, Université de Montpellier II, 14 Décembre 2006.

[9] L. Grisoni, "Vers une simulation physique temps réel multi-modèle," Ecole doctorale de mathématiques, Université Lille I, 09 Décembre 2005.

[10] X. Heurtebise, "Représentation multi résolution et déformation d'objets définis par énumérations spatiales," thèse de doctorat, Université de la méditerranée Aix Marseille II, 15 Octobre 2007.

[11] D. Baraff, "Dynamic simulation of non-penetrating rigid bodies," Thèse de Doctorat, Computer Science Department, Université de Cornell, 1992.

[12] F. Cordier, "Real-time animation of dressed virtual humans, " Thèse de doctorat, Université de Genève, 2004.

[13] S. Benameur, N. E Djedi, "Les Problèmes de Recherche dans la Simulation de Vêtements," JIG - 3èmes Journées Internationales sur l'Informatique Graphique, 2007.

[14] J. Dequidt, L. Grisoni, P. Meseure, C. Chaillou, "Détection de collisions entre objets rigides convexes autonomes," Revue internationnal CFAO et informatique graphique, Volume n°18 (2), 2003.

[15] P. Volino, N. Magnenat Thalmann, "Efficient Self-Collision Detection on Smoothly Discretized Surface Animations using Geometrical Shape Regularity," EUROGRAPHICS'94 Conference, Computer Graphics Forum, 13(3), Oslo, 12-16 Septembre 1994, p 155-166.

[16] P. Volino, N. Magnenat Thalmann, "Collision and Self-Collision Detection: Efficient and Robust Solutions for Highly deformable Surfaces," EUROGRAPHICS workshop on Animation and Simulation, Maastricht, 2-3 Septembre 1995, p 55-65.

[17] S. Cameron, "Enhancing GJK: Computing minimum and penetration distances between convex polyedra," international conference of robotics and automation, Albuquerque, Avril 2007, p 22-24.

[18] G. Van den Bergen, "A Fast and Robust GJK Implementation for Collision Detection of convex Objects," Journal of Graphics Tools, 4(2), 1999, p 7-25.

[19] D. Baraff, A.Witkin, "Dynamic simulation of non-penetrating flexible bodies," SIGGRAPH'92, Computer Graphics, volume 26, Number 2,Chicago, juillet 1992.

[20] D. Baraff, "Analytical Methods for Dynamic Simulation of Non-Penetrating Rigid Body,"SIGGRAPH'89 Conference Proceedings, Computer Graphics, 23(3), Boston, 31 Juillet - 4 août 1989, p 223-232.

[21] D. Baraff, "Fast Contact Force Computation for Non-Penetrating Rigid Bodies," SIGGRAPH' 94 Conference Proceedings, Computer Graphics annual conference series, Orlando, 24 - 29 juillet 1994, p 23-33.

[22] D. Baraff, "Curved Surfaces and Coherence for Non-Penetrating Rigid Body Simulation,"SIGGRAPH'90 Conference Proceedings, Computer Graphics, 24(4), Dallas, 6 Aout 1990, p 19-28.

[23] D. Jovanoski, "The Gilbert – Johnson – Keerthi (GJK) Algorithm," Bachelor Seminar, Fevrier 2008.

[24] A. Fuhrmann, C. Grob, V. Luckas, "Interactive Animation of Cloth including Self Collision Detection " Journal of WSCG, Vol.11, N°1,Fevrier 2003.

[25] Y.J Choi, Y.J. Kim, M.H. Kim, "Self-CD: Interactive Self-collision Detection for Deformable Body Simulation Using GPUs," Asia Sim 2004, LNAI 3398, 2005, pp. 187–196.

[26] Y. Wang, "GPU Based Cloth Simulation on Moving Avatars, " Mémoire de master, Université de Maryland, 2005.

[27] S. Kimmerle, J. Mezger, "Collision Detection for Cloth Simulation," Tutorial1: Simulation of Clothes for Real-time Applications, Published by INRIA and the Eurographics Association, 2004, pp 45-50.

[28] K.J. Choi, H.S. Ko, "Research problems in clothing simulation " Journal of Computer Aided-Design, 37, 2005, pp 585–592.

[29] J. Rodríguez-Navarro, M. Sainz, A. Susín, "GPU Based Cloth Simulation with Moving Humanoids," CEIG'2005, 2005, pp 147-155.

[30] T. Vassilev, B. Spanlang, "Efficient cloth model for dressing animated virtual people," proceedings of Learning to Behave Workshop, Enschede, the Netherlands, 2000, pp 89-100.

[31] E. Kieran, G. Harrison, L. Openshaw, "Cloth Simulation," Master's thesis, Bournemouth University, Poole, UK, 2005.

[32] M Krus, "Maillage Polygonaux et Niveaux de Détails étude bibliographique," Notes et Documents LIMSI, Mai 1997, pp 97 – 10.

[33] H. Charfi, "Amélioration de la modélisation et de la simulation de vêtements en 3D," Thèse de Doctorat de l'Université Paris 6, 2006.

[34] C. Luible, N. Magnenat-Thalmann, "The simulation of cloth using accurate physical Parameters," In Proceedings of the Tenth IASTED International Conference on Computer Graphics and Imaging, 2008, pp 123-128.

[35] C.N.Ngoc, S.Boi, "NonlinearCloth Simulation," Rapport de recherche n° 0123456789, Thème 3 Interaction homme-machine, images, données, connaissances Projet Alcove, INRIA, 2004.

[36] C. Feynman, "Modeling the appearance of cloth," Memoire de Master, Massachusetts University,1986.

[37] D. Terzopoulos, J. Platt, A. Barr, K. Fleischer, "Elastically Deformable Models," SIGGRAPH 87, Anaheim, July, 1987, pp 27-31.

[38] M.Carignan, Y.Yang, N.MagnenatThalmann, D.Thalmann, "Dressing Animated Synthetic Actors with Complex Deformable Clothes," In Proceedings of the 19th annualconference on Computer graphics and interactive techniques, 1992, pp 99 – 104.

[39] F. Zara, "Algorithmes parallèles de simulation physique pour la synthèse d'image : application à l'animation de textiles," thèse de doctorat, INPG université de Grenoble, 2003.

[40] J. Louchet, X. Provot, D. Crochemore, "Evolutionary identification of cloth animation models," In Proceedings of the Eurographics Workshop in Maastricht, The Netherlands, Sep. 95, Springer, pp 44-54.

[41] D. Baraff A. Witkin, "Large Steps in Cloth Simulation," SIGGRAPH 98, Orlando, July1998, pp 19-24.

[42] P. Volino, N. Magnenat Thalmann, "Developing Simulation Techniques for an Interactive Clothing System," In Proceedings. International Conference on Virtual Systems and MultiMedia VSMM '97, IEEE Computer Society1997.

[43] M. Desbrun, P. Schroder, A. Barr, "Interactive animation of structured deformable objects," In Proceedings of the 1999 ACM conference on Graphics interface, 1999.

[44] K. J. Choi H.S. Ko, "Stable but Responsive Cloth," International Conference on Computer Graphics and Interactive Techniques ACM SIGGRAPH 2005.

[45] P.YBurgy, "Intégration de la multirésolution dans un système de déformation masse-ressort," Projet de diplôme, Ecole polythechnique Fédérale de Lausane 2000.

[46] S. Benameur, N. Djedi, "Intégration de la multi-résolution dans un système masse-ressort : application à l'animation de tissu," 4ème Séminaire National en Informatique de Biskra 4, 5, 6 Mai 2004.

[47] S. Hadap, E. Bangerter, P. Volino, N. Magnenat-Thalmann, "Animating Wrinkles on Clothes," Proceedings of the conference on Visualization '99, IEEE Computer Society, 1999, pp 175-182.

[48] D. L. James, K.Fatahalian, "Precomputing Interactive Dynamic Deformable Scenes," Rapport technique CMU-RI-TR-03-33, Robotics Institute, Carnegie Mellon University, September, 2003.

[49] Young-MinKang1,J. H. Choi, H. G. Cho, D.H. Lee, "An efficient animation of wrinkled cloth with approximate implicit integration," The Visual Computer Journal, Springer-Verlag, 2001.

[50] R. Bridson, S. Marino, R. Fedkiw, "Simulation of Clothing with Folds and Wrinkles," Eurographics/SIGGRAPH Symposium on Computer Animation, 2003.

[51] D. Baraff, A. Witkin, M.Kass, "Untangling Cloth," Proceedings of ACM SIGGRAPH 2003,pp 862-870.

[52] P. Volino, M. Courchesne, N. Magnenat Thalmann, "Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects," Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, ACM Press, 1995, pp 137-144.

[53] P. Volino, N. Magnenat Thalmann, S. Carion, D. Thalmann, "The Evolution of a 3D System for Simulating Deformable Clothes on Virtual Actors," IEEE Comp. Graph. & Appl, 16(5), 1996, pp 42-50.

[54] N. Magnenat Thalmann, S. Carion, M. Courchesne, P. Volino, Y. Wu, "Virtual Clothes, Hair and Skin for Beautiful Top Models," Proceedings of the 1996 Conference on Computer Graphics International, IEEE Computer Society, 1996.

[55] F. Cordier, P. Volino, N. Magnenat- Thalmann, "Integrating deformations between bodies and clothes," The journal of visualization and computer animation J. Visual. Comput. Animat. 2001, 12 pp 45-53.

[56] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M. P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, P. Volino, "Collision Detection for Deformable Objects," Computer Graphics Forum 19, 2005, pp 61-81.

[57 T. Vassilev, B. Spanlang, Y. Chrysanthou, "Fast Cloth Animation on Walking Avatars," In Computer Graphics Forum (Proc. of Eurographics), 2001.

[58] X. Provot, " Collision and Self-Collision Handling in Cloth Model Dedicated to Design Garments," In *Graphics Interface '97*,.Canadian Information Processing Society, Canadian Human-Computer Communications Society, May 1997, pp 177-189.

[59] F. Cordier, N. Magnenat-Thalmann, "A Data-driven Approach for Real-Time Clothes Simulation," In Proceedings of the 12th Pacific Conference on Computer Graphics and Applications (PG'04), IEEE computer society, 2004.

[60] S. Kimmerle, "Collision Detection and Post-Processing for Physical Cloth Simulation," Thèse de doctorat, Université de Eberhard-Karls, Tubingue, Allemagne, 2005.

[61] G.Baciu,W.S.K.Wing, "Hardware-assisted self-collision for deformable surfaces," In Proceedings of the ACM symposium on Virtual reality software and technology, SIGGRAPH 2002, pp 129-136.

[62] J. Mezger, S. Kimmerle, O. Etzmuß, "Hierarchical Techniques in Collision Detection for Cloth Animation," Journal of WSCG, Vol.11, No.1, ISSN 1213-6972WSCG 2003, February 3-7, 2003.

[63] J. Mezger, S. Kimmerle, O.Etzmu, "Improved Collision Detection and Response Techniques for Cloth Animation," Technical report, Université de Tubingen, 2002.

[64] N. S. M. Shapri, A. Bade, D.D Aman, "Hierarchy Techniques in Self-Collision Detection for Cloth Simulation," Second International Conference on Machine Vision, 2009. ICMV '09, IEEE Computer Society, Dec 2009, pp 325-329.

[65] A. Fuhrmann, G. Sobottka, C. Grob, "Distance Fields for Rapid Collision Detection in Physically Based Modeling," International Conference Graphicon, Moscow, Russie, 2003.

[66] M. Teschner, B. Heidelberger, M. Muller, D. Pomeranets, M. Gross, "Optimized Spatial Hashing for Collision Detection of Deformable Objects," Technical report, Computer Graphics Laboratory, ETH Zurich, Switzerland, 2003.

[67] G. Baciu, W. S.K. Wong, "Image-Based Collision Detection for deformable Cloth Models," IEEE Transactions on visualization and computer graphics, vol. 10, no. 6, November/December 2004.

[68] S. Jung, M.Hong, M.H. Choi, "An Adaptive Collision Detection and Resolution for deformable Objects Using Spherical Implicit Surface," ICCS 2005, LNCS 3514, Springer-Verlag Berlin Heidelberg, 2005, pp 735-742.

[69] S. Curtis, R. Tamstorf, D. Manocha, "Fast Collision Detection for Deformable Models using Representative-Triangles, In Proceeding of ACM Symposium on Interactive 3D Graphics and Games, 2008.

[70] T. Jund, D. Cazier, J.F. Dufourd, "Système de prédiction pour la détection de collisions dans un environnement déformable," AFIG / Toulouse, IRIT Presse, 2008.

[71] M. Tang, D. Manocha, R. Tong, "Fast Continuous Collision Detection using Deforming Non-Penetration Filters," In Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games, 2010, pp 7-13.

[72] M.Tang, D.Manocha, R. Tong, "MCCD: Multi-core collision detection between deformable models using front-based decomposition," Graphical Models, Volume 72, Issue 2, Mars 2010, pp 7-23.

[73] N. K. Govindaraju, D. Knott, N. Jain, I. Kabul, R, Tamstorf, R. Gayle,M.C. Lin,D. Manocha, "Interactive Collision Detection between Deformable Models using Chromatic Decomposition," In Proceedings of ACM SIGGRAPH 2005,ACM Transactions on Graphics, Volume 24 , Issue 3, pp 991-999.

[74] M. Hutter, A. Fuhrmann, "Optimized Continuous Collision Detection for Deformable Triangle Meshes," In Proc.WSCG '07, Science Press, Plzen, 2007, pp 25-32.

[75] B. He, L. Cheng, "A Quad Tree Based Self-collision Detection Method for Cloth Simulation," Journal of Computers, Vol. 5, N°. 7, JULY 2010.

[76] R. Bridson, R. Fedkiw, and J. Anderson, "Robust Treatment of Collisions, Contact and Friction for Cloth Animation," ACM Transactions on Graphics, ACM SIGGRAPH 2002, 21(3), pp 594-603.

[77] A. Selle, J. Su, G. Irving, and R. Fedkiw, "Robust High-Resolution Cloth Using Parallelism, History-Based Collisions, and Accurate Friction," IEEE Transactions on Visualization and Computer Graphics (TVCG), 2009, 15(2), pp 339-350.

[78] T. Larsson and T. Akenine-Möller, "Collision detection for continuously deforming bodies," In Eurographics 2001, pp 325–333.

[79] M. Moore, J. Wilhelms, "Collision detection and response for computer animation," In Proceedings of ACM SIGGRAPH, 1988, pp 289–298.

[80] D. Terzopoulos, K. Fleischer. "Deformable models," The Visual Computer,4, 1988, pp 306–331.

[81] B. Lafleur, N. Magnenat-Thalmann, and D. Thalmann, "Cloth animation with self-collision detection," In Proceedings of the Conference on Modelling in Computer Graphics,1991, pp 179–187.

[82] B. Eberhardt, A. Weber, and W. Straßer, "A Fast, Flexible Particle-System Model for Cloth Draping," IEEE Computer Graphics and Applications,16(5), 1996, pp 52–59.

[83] J. C. Platt and A. H. Barr, "Constraint methods for flexible models," In Proceedings of ACM SIGGRAPH, 1988, pp 279–288.

[84] G. Van Den Bergen, "Efficient Collision Detection of Complex Deformable Models using AABB Trees," Eindhoven University of Technology 1998.

[85] S. Benameur, N. Djedi, "La Simulation de Vêtements : Etat de l'art et perspectives de recherche," Sixième Séminaire National en Informatique de Biskra SNIB'08, 2008.

[86] D. Hutchinson, M. Preston, T. Hewitt, "Adaptive Reffinement for Mass_Spring Simulations," 7th Eurographics Workshop on Animation & Simulation, Springer.1996.

[87] M. A. Otaduy, Ming C. Lin, "CLODs: Dual Hierarchies for Multiresolution Collision Detection," Eurographics Symposium on Geometry Processing L. Kobbelt, P. Schröder, H. Hoppe (Editors), 2003.

[88] P.M. Hubbar, "Collision detection for interactive graphics," Thèse de doctorat, Université de Brown, Island, 1995.

[89] James T. Klosowski, Martin Held, Joseph S.B. Mitchell, Henry Sowizral, KarelZikan. "Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs," IEEE Transactions on Visualization and Computer Graphics, 1998.

[90] G. Debunne, "Animation multi résolution d'objets déformables en temps-réel Application à la simulation chirurgicale," Thèse de doctorat, Institut National Polytechnique de Grenoble, 2000.

[91] M. Garcia, S. Bayona, P. Toharia, C. Mendoza, "Comparing Sphere-Tree Generators and Hierarchy Updates for Deformable Objects Collision Detection," G. Bebis et al. (Eds.): ISVC 2005, LNCS 3804, Springer-Verlag Berlin Heidelberg, 2005, pp 167–174.

[92] E. Welzl, "Smallest enclosing disks (balls and ellipsoids)," New results and new trends in computer science H. Maurer Edition, Lecture Notes in Computer Science, 1991, pp 359-370.

[93] N. Jain, I. Kabul, N. K. Govindaraju, D. Manocha, M. Lin, "Multi-Resolution Collision Handling for Cloth-like Simulations," Computer Animation and Virtual Worlds Special Issue: CASA 2005, July 2005, Volume 16, Issue 3-4, pp 141–151.

[94] R. Rodriguez, "Détection de collision et localisation de contact des polyèdres déformables en temps- réel dans un environnement virtuel," Rapport DEA Imagerie Vision Robotique. INRIA Rhône-Alpes France, 2003.

[95] O. Nocent, "Animation réaliste de textiles," 5èmes journées *AFIG*, Rennes, 3-5 décembre 1997, pp 227-235.