Université de Strasbourg

École doctorale : **Mathématiques, Sciences de l'Information et de l'Ingénieur**

Unité de recherche : **ICube – UMR 7357**

THÈSE

Présentée par : **Quentin WENDLING**

Soutenue le : **20 décembre 2023**

Pour obtenir le grade de : **Docteur de l'université de Strasbourg**

Discipline/Spécialité : **Informatique**

Couplage géométrie/mécanique pour l'animation d'objets détaillés

Thèse dirigée par :

M. CAZIER David	Professeur des universités, Université de Strasbourg

Rapporteurs :

M. MESEURE Philippe	Professeur des universités, Université de Poitiers
M. JAILLET Fabrice	Maitre de conférences, Université Claude Bernard Lyon 1

Examinateurs :

Mme. BECHMANN Dominique	Professeure des universités, Université de Strasbourg
M. BARTHE Loïc	Professeur, Université Toulouse III
M. COURTECUISSE Hadrien	Chargé de recherches, CNRS – ICube UMR 7357
M. CASTAGNE Nicolas	Maitre de conférences, Grenoble INP

Remerciements

Premièrement, je tiens à remercier le jury de cette thèse en commençant par Philippe MESEURE et Fabrice JAILLET pour avoir accepté de rapporter cette thèse et ensuite Dominique BECHMANN, Loïc BARTHE, Hadrien COURTECUISSE et Nicolas CASTAGNE pour avoir accepté d'assister à cette thèse, j'attends avec grand intérêt leurs retours sur mon manuscrit.

Je remercie également David CAZIER pour m'avoir dans un premier temps accordé sa confiance lors de mon projet de fin de master puis de l'avoir réitéré lors d'un stage et enfin lors de cette thèse et ainsi m'avoir permis de travailler sur cette thématique passionnante. Je le remercie également pour sa patience et sa pédagogie dans son encadrement et avoir su me guider tout au long de cette thèse.

Je remercie également tous les membres de l'équipe IGG, avec qui j'ai eu le plaisir d'échanger et de partager des pauses pendant ces années de thèse, également l'ensemble des doctorants anciens comme nouveaux que j'ai pu rencontrer du début à la fin de cette thèse. En particulier Pascal, Nicolas et Katia avec qui j'ai eu la chance de partager mon bureau et qui m'ont permis d'égayer certaines de mes journées.

Merci également à mes amis et ma famille qui m'auront soutenu volontairement ou non à travers un sourire ou des paroles ou simplement en m'ayant accompagné lors d'une séance de sport.

Je remercie tout particulièrement Gauthier et Charlotte qui, chacun à leurs manières, m'ont accompagné et supporté presque quotidiennement et m'ont permis de rendre ces 4 années de thèse un peu plus belles.

Table des matières

Chapitre 1

Introduction

1.1 Contexte

La manipulation et la représentation d'objets et de scènes complexes dans des environnements virtuels représentent un enjeu majeur dans de nombreux domaines de l'informatique graphique, tels que la visualisation de données, la simulation médicale, les jeux vidéo et la réalité virtuelle. Depuis les travaux fondateurs de Terzopoulos et al. en 1987 [Ter+87], d'importantes recherches ont été menées pour améliorer le réalisme physique des animations générées par les systèmes informatiques, et ont porté notamment sur la représentation des solides déformables et de leurs comportements.

Dans la réalité, ces objets sont définis de manière continue. Pour les manipuler numériquement et pour estimer les grandeurs physiques modélisant leur comportement, il est nécessaire de les discrétiser. Une approche courante consiste à utiliser une décomposition cellulaire de leur intérieur, définissant un ensemble de sommets, arêtes, faces et volumes. La finesse de cette discrétisation influe directement sur la qualité de la représentation et, par conséquent, sur la précision des grandeurs physiques calculées.

Les besoins en matière de simulation d'objets déformables ont évolué au fil des années. Il est devenu courant de vouloir représenter des objets intégrant des détails géométriques très fins, de pouvoir appliquer des changements topologiques tels que des déchirures ou des fractures, tout en garantissant des performances en temps réel des systèmes d'animation. Cela nécessite de trouver un équilibre entre la précision et la rapidité des calculs effectués.

Dans le domaine de l'informatique graphique, où le rendu visuel est crucial et où l'on attend des résultats riches en détails, la performance souhaitée est souvent obtenue en plongeant les objets détaillés dans des maillages grossiers, dont les sommets forment les degrés de liberté du modèle physique. Pour atteindre les performances souhaitées, la simulation est effectuée sur les maillages grossiers, dont les positions sont interpolées pour déterminer les déformations des objets détaillés.

C'est le cas de la méthode de Nesme et al.[Nes+09], qui propose d'intégrer chaque objet dans une structure englobante tout en préservant sa topologie. Rivers et al.[RJ07] ont également proposé une méthode d'animation dans laquelle ils intègrent les objets

animés dans une grille régulière, ce qui permet l'utilisation de méthodes numériques efficaces pour la simulation physique tout en réduisant le niveau d'échantillonnage de la scène. Cependant, dans ces travaux, la précision de la simulation reste directement liée à la résolution des maillages grossiers. Celle-ci est fixée avant la simulation, lors d'un prétraitement, et reste constante durant la simulation. Ainsi, une structure d'intégration fine devient nécessaire pour obtenir la qualité visuelle souhaitée, ce qui limite les avantages réels de ces approches historiques.

Pour faire face à cette problématique, des approches dites multirésolutions ont été proposées. Au lieu de représenter un objet par un unique maillage, il est représenté par une hiérarchie de maillages de plus en plus fins. Cette hiérarchie peut être construite soit de manière ascendante, en augmentant progressivement la résolution d'un maillage grossier, soit de manière descendante, en réduisant progressivement la résolution d'un maillage fin. Ces deux approches peuvent être utilisées conjointement.

Bien que de nombreux modèles multirésolutions aient déjà été proposés, chacun ayant ses caractéristiques et ses avantages, aucun d'entre eux ne permet de gérer des parcours adaptatifs simultanés et efficaces des différents niveaux de résolution pour permettre un contrôle précis et local du niveau de détail des objets manipulés. Cela signifie qu'actuellement, il est nécessaire de parcourir l'ensemble de la résolution, ce qui pose problème dans des situations où les interactions sont très localisées et où les degrés de liberté peuvent être concentrés de manière très locale pour rester dans un budget de temps de calcul compatible avec le temps réel.

Ces dernières années, les méthodes adaptatives ont pris le pas sur les approches historiques, comme le montre [Man+17]. Elles permettent d'insérer localement et dynamiquement des degrés de liberté dans la simulation, avec pour objectif d'adapter les modèles physiques à l'environnement de simulation. Différentes méthodes d'adaptation ont été proposées. Les premières sont basées sur des structures hiérarchiques, notamment les octrees [NFP06 ; DGW11]. L'adaptation locale est réalisée en activant ou désactivant les nœuds de l'octree à différentes profondeurs. Ensuite, des parcours adaptatifs des octrees sont mis en œuvre pour restreindre les degrés de liberté utilisés à ceux qui sont actifs. Plusieurs méthodes basées sur les octrees ont été recensées dans la littérature, se distinguant principalement par les critères d'adaptation utilisés. Elles nécessitent toutes de coupler chaque sommet du maillage fin à ceux du modèle grossier, ce qui est généralement réalisé lors d'une phase de prétraitement, ce qui rend difficile la prise en compte des changements topologiques au cours de la simulation.

1.2 Contributions

Comme nous venons de le voir, l'utilisation de maillages adaptatifs revêt une importance capitale pour la simulation et l'animation d'objets hautement détaillés. Cependant, les modèles multirésolutions actuels ne proposent pas de parcours adaptatifs efficaces au sein de différents niveaux de résolution. Les cartes combinatoires multirésolutions ont déjà fait preuve de leur efficacité dans le domaine de la modélisation géométrique. Bien qu'elles ne permettent pas, dans leur implantation actuelle, de parcours adaptatifs, nous

sommes convaincus qu'elles offrent un cadre pertinent pour la gestion de maillages adaptatifs, supportant des changements topologiques, dans un contexte multirésolution.

Dans cette thèse, nous présentons une extension des cartes combinatoires qui introduit ce que nous appelons des *vues multirésolutions adaptatives*. Ces vues permettent un parcours adaptatif des niveaux de résolution et offrent un mécanisme de construction de hiérarchies de vues pour combiner finement et efficacement différentes adaptations. Nous démontrons l'efficacité des vues multirésolutions en proposant divers benchmarks pour mesurer et illustrer leur efficacité sur des jeux d'opérations topologiques représentatifs des besoins d'un moteur d'animation interactif.

Dans une seconde partie, nous appliquons les vues multirésolutions au développement d'un framework d'animation multirésolution générique. L'utilisation des vues permet de contrôler précisément le nombre de degrés de liberté utilisés au sein de la simulation, tout en offrant la possibilité de combiner différents critères d'adaptation et d'ajuster la répartition des degrés de liberté au sein des discrétisations manipulées.

Pour démontrer la pertinence de notre framework, nous détaillons l'implantation de trois modèles physiques de nature différente. Il s'agit de l'algorithme du *Shape Matching* qui détermine les contraintes de déplacements à partir des positions des sommets, de sa variante basée sur la physique qui utilise une approximation du gradient de déformation dans un modèle néo-Hookéen , et enfin de la méthode des *Smoothed-Particle Hydrodynamics (SPH)* qui estime le gradient de déformation à partir d'une discrétisation des objets sous forme de particules associées à des fonctions noyaux.

Nous explorons également différents critères d'adaptation permettant de déterminer où et quand ajouter ou retirer des degrés de liberté dans la simulation. Cela inclut des critères géométriques basés sur les déformations subies par les éléments, des critères mécaniques exploitant le calcul des tenseurs de déformation, et enfin des critères multirésolution qui comparent les résultats obtenus en calculant les déformations à différents niveaux de résolution. Pour finir, nous présentons différentes méthodes de propagation des déformations au sein de la hiérarchie multirésolution.

Ces approches sont évaluées dans différents scénarios permettant d'illustrer les possibilités de l'approche proposée. Le premier scénario propose la simulation de contact entre un objet et un instrument contrôlé par l'utilisateur incluant une adaptation de l'objet dans les zones de contact. Le second scénario montre les interactions d'un objet avec un environnement virtuel et inclue une adaptation d'objet dépendant de ses déformations. Le troisième scénario propose une adaptation de l'objet en fonction de l'angle de vue d'une caméra en mouvement. Le denier scénario illustre la flexibilité des vues multirésolutions en proposant un algorithme de découpe et de simulation de déchirure intégré au framework proposé.

1.3 Plan de la thèse

Ce manuscrit de thèse est structuré en deux parties. La première partie regroupant les chapitres 2 et 3, se concentre sur l'état de l'art et les contributions de cette thèse dans le domaine des modèles topologiques multirésolutions. La deuxième partie, incluant les

chapitres 4 et 5, se penche sur l'utilisation du modèle des vues multirésolutions pour l'animation adaptative d'objets déformables.

Dans le chapitre 2, nous présentons les modèles topologiques disponibles dans la littérature, avec une attention particulière pour les modèles multirésolutions et leurs applications. Nous présentons également de manière détaillée le modèle des cartes combinatoires et sa version multirésolution.

Dans le chapitre 3, nous présentons nos contributions, en mettant en avant les vues multirésolutions, un modèle topologique basé sur les cartes combinatoires multirésolutions. Nous introduisons la notion de *visibilité* pour permettre un parcours adaptatif des niveaux de résolution. Nous explorons également un mécanisme permettant de combiner plusieurs vues. Enfin, nous décrivons un mécanisme de séparation topologique, préparant le terrain aux opérations de découpe. Ce chapitre se clôture par une série de tests visant à démontrer les performances et la pertinence de ce modèle topologique.

Dans le chapitre 4, nous abordons les concepts fondamentaux de l'animation d'objets déformables. Nous présentons les modèles mécaniques couramment utilisés pour régir le comportement de ces objets. Ensuite, nous explorons les différentes catégories de méthodes disponibles dans la littérature pour permettre une simulation adaptative de scènes complexes, c'est-à-dire pour adapter la répartition des dégrées de liberté pour garantir des performances en temps réel avec la précision souhaitée. Enfin, nous examinons les différentes méthodes de découpe et de déchirure disponibles dans l'état de l'art, ainsi que leurs limitations dans un contexte adaptatif.

Le chapitre 5 présente les expérimentations menées pour démontrer l'intérêt vues multirésolutions adaptatives. Nous décrivons les mécanismes mis en place pour propager les déformations au sein de la hiérarchie multirésolution. Nous détaillons l'adaptation de trois modèles mécaniques pour leur utilisation dans un contexte adaptatif. Nous nous penchons ensuite sur les différents critères d'adaptation développés. Nous présentons enfin une évaluation de notre framework dans divers scénarios. Ce chapitre se clôture par une présentation d'un algorithme de découpe multirésolution et une discussion sur les perspectives qu'offre notre modèle.

Enfin, dans le dernier chapitre, nous concluons en rappelant les contributions et résultats obtenus, en évoquant leurs limitations et les difficultés rencontrées. Nous envisageons quelques prolongements possibles dans le domaine de la simulation interactive.

Première partie

Modèles topologiques adaptatifs pour l'animation

Chapitre 2

État de l'art des modèles topologiques

2.1 Modèles monorésolutions

Dans le monde réel, les objets sont définis de manière continue et donc constitués d'une infinité de points. Pour pouvoir les représenter et les manipuler à l'aide de l'outil informatique qui est intrinsèquement fini, il convient de les discrétiser. Il existe essentiellement deux manières de représenter des objets continus. La première consiste à les représenter de manière *implicite* à l'aide d'un ensemble fini de fonctions de $\mathbb{R}^3 \rightarrow \mathbb{R}$ et de paramètres. L'intérieur d'un objet modélisé par une telle fonction est constitué des points de l'espace pour lesquels la fonction à une valeur négative. Le bord de l'objet est constitué des points où la fonction est nulle. Des telles représentations permettent difficilement de contrôler la topologie et la discrétisation des objets manipulés et ne seront pas considérées plus avant dans ce manuscrit.

Nous choisissons ici les méthodes consistant à définir chaque objet comme une partition de l'espace. Les objets sont décomposés en cellules de différentes dimensions. En 3D, un objet est ainsi décomposé en sommets, arêtes, faces et volumes. Chaque cellule est définie par son bord, lui-même décomposé en cellules de dimensions inférieures. Les différentes cellules possèdent des relations d'incidence pour des cellules de dimension différentes et d'adjacences pour des cellules de même dimension. Les cellules définissent ce qu'on appelle aussi un maillage de l'espace voir Figure 2.1). Il est important de noter que la discrétisation d'un objet sous forme de maillage est cruciale pour le calcul numérique, notamment pour la méthode des éléments finis qui utilise cette discrétisation pour résoudre des équations différentielles.

Dans cette thèse, nous nous intéresserons principalement aux représentations volumiques. De nombreuses applications utilisent de tels maillages et l'accès aux différentes informations topologiques de manière optimisée est un problème critique dans de nombreuses applications. Ainsi, la mise au point de structures de données optimisées pour ces usages est nécessaire pour le développement de moteur d'animation interactif. Dans la suite, nous présentons les principales structures de données permettant de modéli-

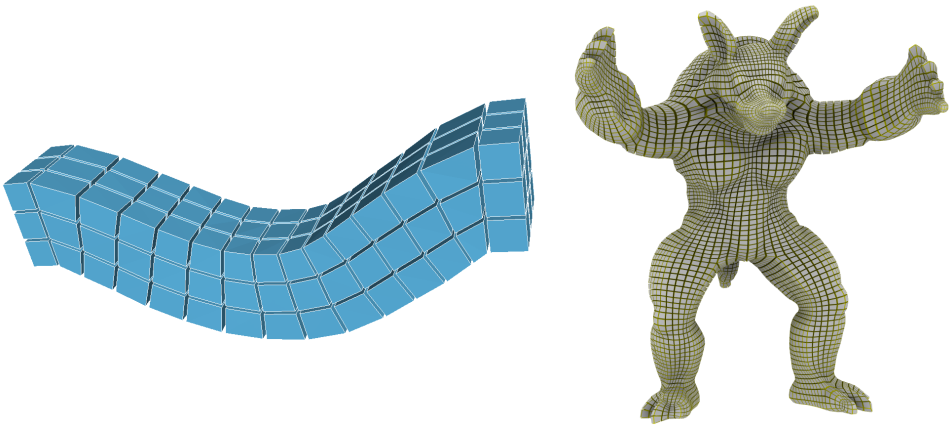

FIGURE 2.1 – Deux maillages volumiques hexaédriques : (à droite) la simulation d'une poutre soumise à la gravité ; (à gauche) l'animation d'un Armadillo.

ser des décompositions cellulaires, en partant des modèles non structurés, pour aller vers les modèles les plus structurés permettant de récupérer l'ensemble des informations topologiques existant. Nous nous intéressons particulièrement au modèle des **cartes combinatoires** qui constituent la base des travaux de cette thèse.

2.1.1 Modèles topologiques non structurés

Une première méthode de discrétisation consiste à représenter des objets préalablement décomposés en cellules liées entre elles par des relations d'incidence et d'adjacence. On parle de complexes cellulaires (voir figure 2.2. En dimension n, un complexe cellulaire est composé d'un ensemble de k-cellules avec $k \in [0, n]$. En dimension 3, les 0-cellules sont des sommets, les 1-cellules sont des arêtes, les 2-cellules sont des faces et les 3-cellules sont des volumes. Toute k-cellule pour $k \geq 1$ possède un bord constitué de $(k-1)$-cellules. Une 0-cellule ne possède pas de bord. À partir de cette notion de bord, les relations de voisinage entre cellules sont définies. Une $(k-1)$-cellule c_1 est dite incidente à une k-cellule c_2 si c_1 appartient au bord de c_2, inversement c_2 est également incidente à c_1. Deux k-cellules sont dites adjacentes si elles sont incidentes à au moins une même $(k-1)$-cellule, c'est-à-dire que leurs bords ont au moins une cellule en commun.

Il est important de noter que cette décomposition cellulaire modélise uniquement la topologie d'un objet. Pour obtenir une représentation concrète, il faut lui ajouter des informations de plongement. Le plongement peut définir la géométrie de l'objet pour laquelle chaque sommet est représenté par un point de $\mathbb{R}^3$, chaque arête par un segment ou une courbe et chaque face par une surface. À noter que la dimension topologique n'est pas nécessairement corrélée à la dimension du plongement. Il est courant, par exemple, de plonger une surface dans $\mathbb{R}^3$. À noter également que le plongement n'est pas exclusivement une information géométrique, il peut également porter des grandeurs

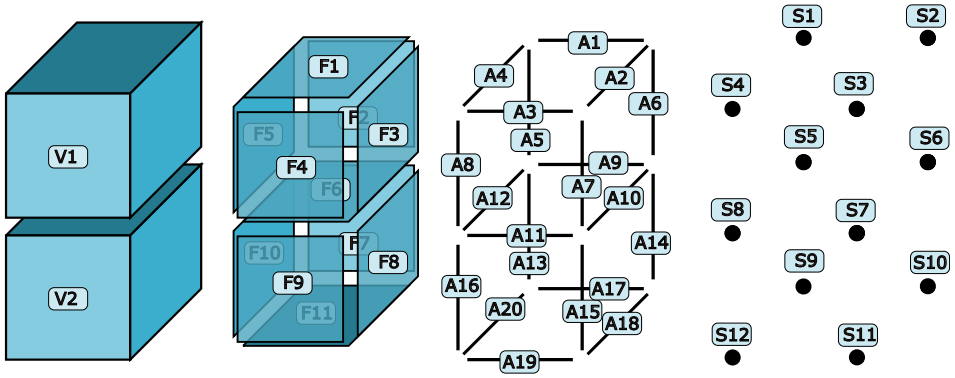

FIGURE 2.2 – Décomposition cellulaire deux hexaèdres partageant une face commune (de gauche à droite : décomposition en volumes, faces, arêtes et sommets)

physiques pour la modélisation de comportements mécaniques.

Graphes d'incidence

Une première approche consiste à représenter explicitement la décomposition cellulaire des objets sous forme de *graphes d'incidence* [Ede87]. Un graphe d'incidence est un graphe orienté dans lequel les nœuds du graphe représentent les cellules de l'objet et les arêtes les relations d'incidences entre les cellules. Les seules relations d'incidence représentées sont celles entre les cellules de dimensions successives (entre sommets et arêtes, arêtes et faces, faces et volumes...). Les autres relations d'incidence peuvent être récupérées en parcourant le graphe, de même pour les relations d'adjacence, deux cellules étant adjacentes si elles sont incidentes à une même cellule.

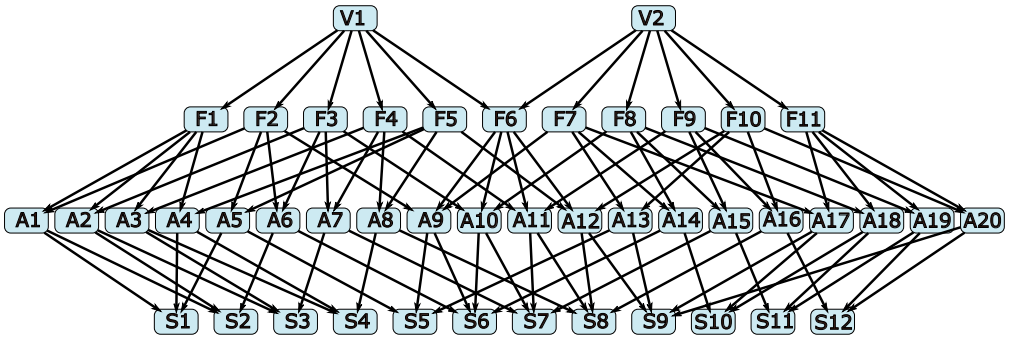

FIGURE 2.3 – Graphe d'incidence de l'objet présenté sur la figure 2.3

Il est possible d'accéder à l'ensemble des $(k-1)$-cellules d'une k-cellule directement en parcourant les fils d'un nœud du graphe. En revanche, l'accès aux k-cellules incidentes à une $(k-1)$-cellule est plus difficile. Il faut parcourir toutes les k-cellules pour obtenir

cette liste. Une solution consiste à enrichir le graphe avec des arêtes d'orientation inverse reliant les $(k-1)$-cellules à leurs k-cellules incidentes.

Cette représentation, bien que simple et utilisable pour représenter un grand nombre d'objets, pose cependant un certain nombre de problèmes. Le premier problème est que deux objets ayant une topologie différente peuvent être représentés par le même graphe d'incidence. Cela vient du fait qu'il n'y a pas de relation d'ordre entre les cellules d'une même dimension lors de la construction d'une cellule de dimension supérieure, ce qui ne permet pas, par exemple, de définir l'orientation des faces de l'objet. Deux objets ayant des faces orientées différemment peuvent ainsi être représentés par un même graphe d'incidence.

Graphes d'incidence indicés

Une solution à ce problème est l'utilisation de graphes d'incidence indicés, notamment utilisée dans le format de fichier OBJ, ou dans certaines bibliothèques de simulation comme SOFA [All+07]. Cette structure est utilisée dans de nombreuses applications interactives du fait que sa structure à base de tableaux permette des accès simples aux informations topologiques et que celles-ci soient alignées en mémoire.

Dans un graphe d'incidence indicé chaque cellule est représentée par un indice dans un tableau propre à sa dimension. Il y a donc un tableau par dimension de cellules représentées. Chaque ligne de ces tableaux stocke les informations concernant une cellule en listant les indices des cellules incidentes ou adjacentes, ainsi que les informations de plongement. Par exemple pour représenter une surface, un premier tableau contient les sommets et leur position dans $\mathbb{R}^3$, un second tableau contient les faces définies comme des listes ordonnées d'indices de sommets. Les arêtes sont définies implicitement par les indices de deux sommets successifs d'une face.

Bien que plus performant que les graphes d'incidence, le format OBJ a d'autres lacunes comme le fait que l'accès à des informations topologiques non explicitement stockées et très coûteux car il nécessite de parcourir l'ensemble des cellules du graphe. Par exemple pour récupérer l'ensemble des sommets adjacents par les arêtes à un sommet donné v, il faut parcourir l'ensemble des faces pour récupérer la liste des arêtes contenant v, puis renvoyer la liste des sommets reliés à v par une arête, tout en prenant soin d'éviter les doublons car, excepté au bord de la surface, toute arête est partagée par deux faces.

Une solution pourrait être de stocker explicitement l'ensemble des relations nécessaires à l'application. Cette solution soulève d'autres difficultés. Premièrement les relations d'incidence dépendent du degré des cellules et ne sont donc pas de tailles fixes et vont donc demander un stockage dynamique. Deuxièmement, il est très difficile de définir des contraintes d'intégrité sur une telle structure de données. Le développement d'opérations topologiques complexes reste sources d'erreurs fréquentes.

2.1.2 Modèles topologiques structurés

Nous allons voir qu'en limitant la classe des objets représentables aux quasi-variétés, il est possible de définir des structures de données compactes et bien plus efficaces.

Commençons par définir la notion de variété topologique illustrée par la Figure 2.4.

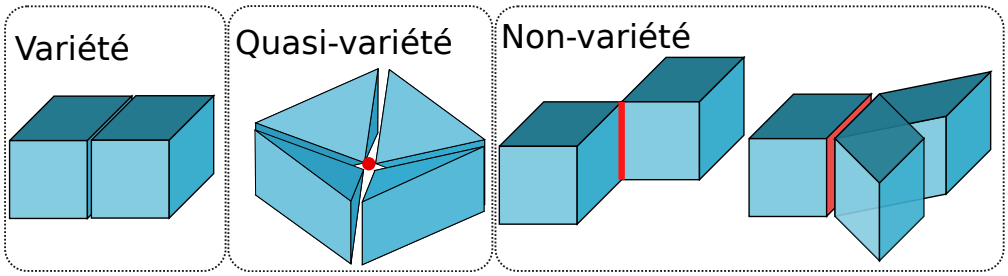

FIGURE 2.4 – De gauche à droite : une variété, une quasi-variété où le sommet central n'est pas homéomorphe à une sphère, deux non-variétés avec respectivement 4 faces incidentes à une arête et 3 volumes incidents à une face

Définition 1 *Une variété de dimension n est un objet où le voisinage de tout point de l'objet est homéomorphe à une sphère de dimension n.*

Définition 2 *On appelle une quasi-variété de dimension n un objet constitué d'un ensemble de n-cellules, adjacentes entre elles le long de $(n-1)$-cellules en respectant la contrainte que toute $(n-1)$-cellule est incidente à au plus deux n-cellules.*

Modèles basés sur les arêtes

Les premiers modèles structurés étaient dédiés à la représentation de surfaces. Ils étaient construits à partir de la notion d'arête. Dans une variété fermée de dimension 2, les arêtes ont pour propriété d'être toujours incidentes à exactement deux faces et deux sommets. Leurs relations d'incidences sont donc faciles à stocker.

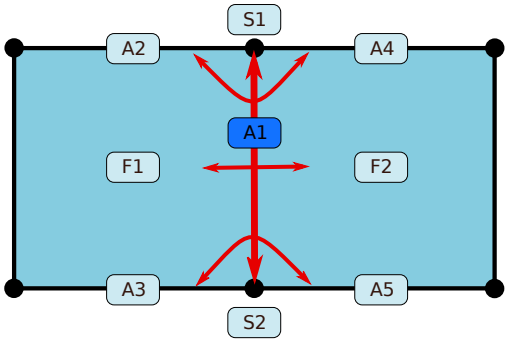

FIGURE 2.5 – Modèle des arêtes ailées : les pointeurs de l'arête A1

Une des structures de données la plus courante est la structure des arêtes ailées [Bau75 ; Gla91]. Dans cette structure, toutes les cellules (arêtes, sommets et faces) sont

explicitement représentées et un certain nombre d'attributs sont associés à chaque cellule. Pour les sommets, une référence à une arête incidente est stockée ; pour les faces, il s'agit d'une référence à une arête incidente ; pour les arêtes, il s'agit des références aux deux sommets qui lui sont incidents, aux deux faces qui lui sont incidentes, ainsi que des références aux deux arêtes suivantes et aux deux arêtes précédentes. Cette structure illustrée dans la figure 2.5 décrit la topologie d'un objet. Il est possible d'ajouter différents plongements aux cellules représentées.

Cette structure contient toutes les relations d'incidence et d'adjacence de l'objet, même pour un maillage polygonal quelconque. Mais comme certaines informations sont redondantes, les parcours de voisinage nécessitent de distinguer différents cas. Par exemple si l'on veut parcourir les sommets adjacents par les arêtes à un sommet, il faut savoir pour chaque arête si le sommet central est le premier ou le deuxième sommet de l'arête.

Modèles basés sur les demi-arêtes

Pour diminuer la redondance et éviter de devoir distinguer les cas, la structure des demi-arêtes propose de remplacer les arêtes ailées, qui sont par définition non orientées, par deux demi-arêtes orientées [Män87 ; Ket99].

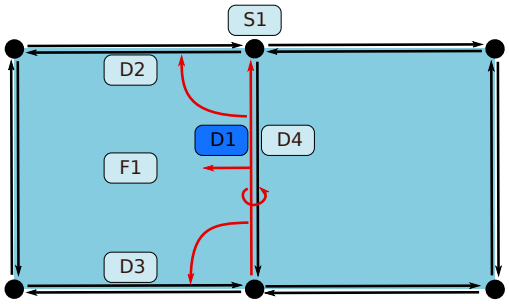

FIGURE 2.6 – Modèle des demi-arêtes : les pointeurs de la demi-arête D1

Ces demi-arêtes ne stockent ainsi plus qu'une référence à un sommet, à une face, et aux arêtes suivante et précédente, ainsi qu'une référence à la deuxième demi-arête. Cette structure illustrée dans la figure 2.6 décrit, comme la précédente, l'intégralité des informations topologiques de l'objet avec moins de redondance. Elle est très utilisée en informatique graphique, notamment pour la modélisation de surfaces. Elle permet de parcourir facilement les voisins d'une cellule donnée, ainsi que de naviguer dans la topologie de l'objet. Elle est également utilisée pour l'implantation de certaines opérations topologiques telles que la fusion ou la séparation de faces.

Notons ici que l'introduction d'un élément abstrait, les demi-arêtes, qui n'existe pas explicitement dans la décomposition cellulaire, permet de simplifier les données stockées et d'optimiser les parcours topologiques. Cette approche a été exploitée pour représenter efficacement des objets de dimension supérieure à l'aide des cartes généralisées, puis des cartes combinatoires, comme nous allons le voir dans la prochaine section.

2.1.3 Cartes généralisées

La définition des cartes généralisées de dimension n, notées n-G-cartes, est basée sur la notion de tuples de cellules. Un tuple de cellules pour un objet de dimension n et un ensemble ordonné de cellules $(c_n, c_{n-1}, ..., c_0)$ de dimensions décroissantes tel que pour tout $0 < i \leq n$ la cellule c_i et incidente à la cellule c_{i-1}. Chaque tuple représente un chemin dans le graphe d'incidences entre une cellule de dimension 0 et une cellule de dimension n.

Les relations d'adjacences entre tuples sont définies ainsi : deux tuples sont i-adjacents s'ils ne diffèrent que par une cellule de dimension i. Notons que dans une quasi-variété sans bord ces relations de i-adjacence lient toujours les tuples deux par deux et que dans une quasi-variété avec bord, les tuples du bord sont n-adjacent à aucun autre tuple.

La définition des n-G-cartes repose sur une entité abstraite appelée brin qui modélise un tuple de cellules. Chaque brin représente un tuple et chaque tuple est représenté par un brin. Les brins sont ensuite liés entre eux par $n + 1$ relations α_i représentant les i-adjacences entre tuples.

Dans une quasi-variété, nous avons vu que la i-adjacence est une relation qui lie les tuples deux par deux, sauf sur le bord, les relations α_1 sont des involutions, c'est-à-dire que pour tout brin d : $\alpha_i(\alpha_i(d)) = d$ et si d est un brin du bord, alors il est en relation avec lui-même, c'est que dire que $\alpha_i(d) = d$. Cela nous amène à la définition des cartes généralisées de dimension n :

Définition 3 *Une carte généralisée de dimension n ou n-G-carte est un* $(n + 2)$*-uplet* $(B, \alpha_0, ..., \alpha_n)$ *où :*

- B *un ensemble fini de brins*
- $\forall i : 0 \leq i \leq n$, α_i *est une involution sur* B
- $\forall i, j : 0 \leq i < i + 2 \leq j \leq n, \alpha_i \circ \alpha_j$ *est une involution*

La contrainte correspondant au troisième point de cette définition garantit que l'objet modélisé est bien une quasi-variété. Pour les surfaces, elle impose le fait qu'une arête est partagée par uniquement deux faces. Pour les volumes, elle impose qu'ils soient adjacents par des faces complètes.

Cette représentation abstraite des n-G-cartes définit complètement les cellules et leurs relations d'incidence et d'adjacence. Une i-cellule est définie par l'ensemble des brins atteignable depuis un brin donné en appliquant successivement les relations α_j avec $j \neq i$. Cela correspond formellement à l'orbite du brin noté $< \alpha_0, ..., \alpha_{i-1}, \alpha_{i+1}, ..., \alpha_n >$. Ainsi dans le cas d'une 3-G-Cartes les sommets correspondent aux orbites $< \alpha_1, \alpha_2, \alpha_3 >$, les arêtes aux orbites $< \alpha_0, \alpha_2, \alpha_3 >$, les faces aux orbites $< \alpha_0, \alpha_1, \alpha_3 >$ et les volumes aux orbites $< \alpha_0, \alpha_1, \alpha_2 >$. Il est également possible de définir d'autres types de cellules en utilisant moins de relations dans les orbites c'est notamment le cas de coins de volumes $< \alpha_1, \alpha_2 >$ et coins de face $< \alpha_1 >$ qui peuvent être utiles pour stocker des informations notamment physiques.

Pour résumer cette partie, nous avons défini les n-G-cartes comme une structure constituée d'un ensemble d'entités abstraites appelées brins reliées par des involutions

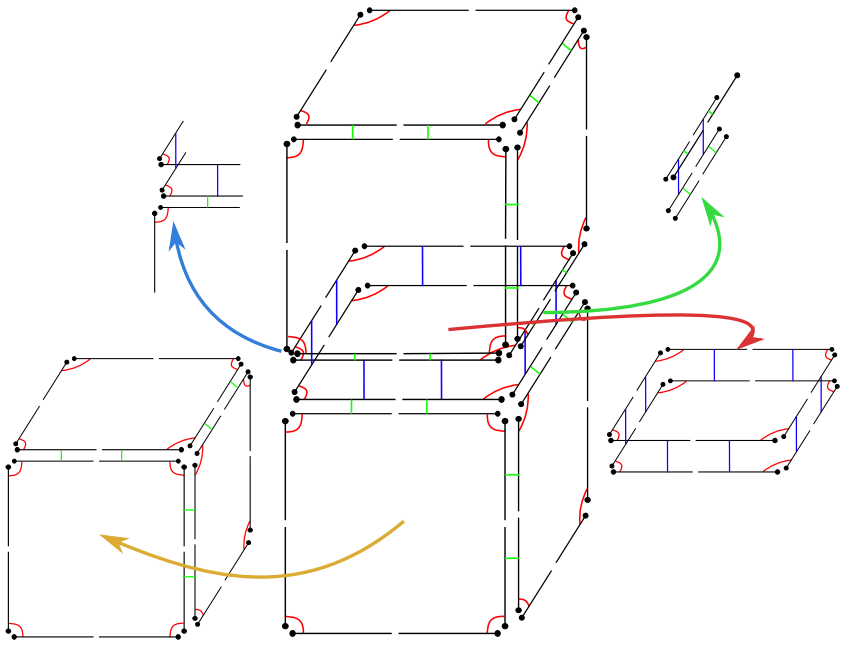

FIGURE 2.7 – Carte généralisée correspondant à la décomposition cellulaire de la figure 2.2. Pour un besoin de visibilité seuls les brins des faces visibles sont représentés. Les relation α_0, α_1, α_2 et α_3 entre les brins sont représentés respectivement : par un espace blanc entre deux brins face à face, un lien rouge, un lien vert et un lien bleu. Quatre orbites de cellules sont représentés : 0-cellule ou sommet (haut gauche), 1-cellule ou arête (haut droite),2-cellule ou face (bas droite),3-cellule ou volume (bas gauche).

α_i, ainsi qu'un ensemble de contraintes sur ces relations. Cette structure permet un accès compact et optimisé aux informations topologiques de l'objet modélisé. Il est possible de démontrer que toute décomposition cellulaire d'une quasi-variété de dimension n est représentable par une n-G-carte.

2.1.4 Cartes combinatoires

Les n-G-cartes modélisent des quasi-variétés orientables ou non. Lorsque l'on manipule uniquement des objets orientables, leur usage peut alourdir les traitements car il faut maintenir l'information d'orientation ce qui peut être complexe en raison de son caractère global. Dans de nombreux cas, il est plus avantageux d'utiliser une structure spécialisée pour représenter des quasi-variétés orientables : les cartes combinatoires.

Pour arriver à la définition des cartes combinatoires, il suffit de déterminer l'orientation d'une n-G-carte à l'aide d'un algorithme de coloriage binaire en appliquant la règle suivante : un brin d'une couleur donnée ne peut être relié qu'à un brin de l'autre couleur. L'algorithme de coloriage consiste à assigner une couleur à un brin choisi au hasard, puis à colorier les autres brins en suivant cette règle, voir la figure 2.8.

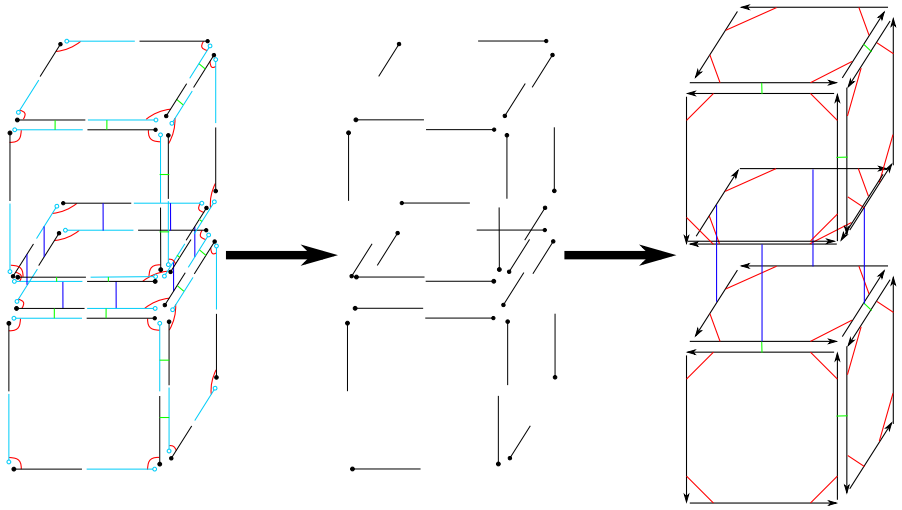

FIGURE 2.8 – Coloriage de la G-carte de la figure 2.7. De gauche à droite : coloriage des brins de la carte, conservation des brins noir, passage à une carte combinatoire. Sur la carte combinatoire à gauche de la figure les relations ϕ_1 (rouge) ,ϕ_2 (vert) et ϕ_3 (bleu) sont représentées.

Si la G-carte est orientable les brins se répartissent en deux ensembles B^1 et B^2 de même cardinal représentant chacun une orientation possible de l'objet. Formellement, il est possible d'obtenir l'ensemble des brins possédant une même orientation en calculant l'orbite $< \alpha_1 \circ \alpha_0, ..., \alpha_n \circ \alpha_0 > (d)$, en partant d'un brin quelconque d. L'ensemble des brins d'orientation opposée est obtenu en calculant l'orbite $< \alpha_1 \circ \alpha_0, ..., \alpha_n \circ \alpha_0 > (\alpha_1(d))$.

Dans une G-carte orientable, les ensemble B1 et B2 représentent exactement la même structure topologique. Il est possible de supprimer la redondance en suppriment un des deux ensembles de brins. On obtient ainsi une modèle deux fois plus compact représentant une des orientations de la G-carte ce qui nous conduit à la définition des cartes combinatoires :

Définition 4 *Une carte combinatoire de dimension n, ou n-carte, est définie par un $(n+1)$-uplet $(B, \phi_1, ..., \phi_n)$ tel que :*

- *B un ensemble fini de brins*
- *ϕ_1 est une permutation sur B*
- *$\forall i : 1 < i \leq n$, ϕ_i est une involution de B dans B*
- *$\forall i, j : 1 \leq i < i+2 \leq j \leq n, \phi_i \circ \phi_j$ est une involution*

Comme pour les cartes généralisées, une cellule de dimension i supérieure à 1 est définie par l'orbite $< \phi_1, ..., \phi_{i-1}, \phi_{i+1}, ..., \phi_n >$. Par contre, un sommet est défini par l'orbite $< \phi_1 \circ \phi_2, ..., \phi_1 \circ \phi_n >$. Cette définition des cartes combinatoires est appelée représentation duale. Il existe également une représentation primale de cartes :

Définition 5 *Une carte combinatoire primale de dimensions n est définie par un* $(n+1)$*-uplet* $(B, \alpha_0, ..., \alpha_{n-1})$ *tel que :*

- B *un ensemble fini de brins*
- α_{n-1} *est une permutation sur* B
- $\forall i : 0 \leq i \leq n-2$, α_i *est une involution de* B *dans* B
- $\forall i, j : 0 \leq i < i+2 \leq j \leq n-1, \alpha_j \circ \alpha_i$ *est une involution*

Ces deux représentations étant duales l'une de l'autre, il est possible d'exprimer les α_i à partir des ϕ_i avec $\alpha_0 = \phi_n$ et $\forall i : 1 \leq i \leq n-1$, $\alpha_i = \phi_i \circ \phi_n$. Et inversement les ϕ_i peuvent être exprimés à travers les α_i avec $\phi_n = \alpha_0$ et $\forall i : 1 \leq i \leq n-1, \phi_i = \alpha_i \circ \alpha_0$.

Les cellules de la représentation primale sont définies comme pour la représentation duale en remplaçant les ϕ_i par les α_i.

2.1.5 Hypercartes

Les cartes combinatoires et les cartes généralisées partagent une structure commune. Elles sont définies par un ensemble de brins muni d'un ensemble de permutations. La différence entre ces deux modèles réside uniquement dans les contraintes imposées à ces permutations.

Il peut être intéressant de se doter d'un modèle plus générique défini par un ensemble de brins muni d'un ensemble de permutations sans contrainte particulière. Il s'agit du modèle des *hypercartes combinatoires*.

Définition 6 *Une hypercarte de dimension n est définie par un* $(n+1)$*-uplet noté* $(B, \phi_0, ..., \phi_{n-1})$ *tel que :*

- B *un ensemble fini de brins*
- $\forall i : 0 \leq i \leq n-1$, ϕ_i *est une permutation de* B *dans* B

Ce modèle permet de définir les n-cartes à partir d'hypercartes de dimension n et les n-G-cartes à partir d'hypercartes de dimension n+1, en ajoutant les contraintes d'intégrité correspondantes. Le modèle des hypercartes sera utilisé par la suite pour définir formellement les cartes combinatoires multirésolutions.

2.1.6 Plongement

Les modèles de cartes généralisées et de cartes combinatoires permettent de représenter la topologie de quasi-variétés de dimension n, c'est-à-dire leur décomposition en cellules ainsi que les relations d'adjacence et d'incidence entre ces cellules. Pour exploiter ces objets dans un contexte applicatif, il est nécessaire de pouvoir plonger les cellules, c'est-à-dire de leur associer divers attributs. Il s'agit par exemple associer une position dans $\mathbb{R}^3$ à chaque sommet, une couleur à chaque face ou une masse à chaque volume.

Afin d'assurer la cohérence des plongements, tous les brins d'une cellule doivent être plongés sur les mêmes attributs. La solution communément utilisée consiste à attribuer un indice à chaque cellule. Cet indice est utilisé pour associer un brin à une cellule. Les attributs des cellules sont alors stockés dans des tableaux exploitant cette indexation.

La structure de carte est ainsi enrichie d'un ensemble de $n+1$ fonctions $pl_i : B \to \mathbb{N}$ qui associent chaque brin à l'indice de la cellule de dimension i à laquelle il appartient. Afin d'assurer la cohérence de la carte, une contrainte est ajoutée sur les fonctions pl_i : $\forall d, d' : d' \in cell_i(d) \Leftrightarrow pl_i(d) = pl_i(d')$. Cette contrainte stipule que si deux brins appartiennent à la même cellule, alors ils sont associés au même indice, et inversement, que si deux brins sont associés au même indice, alors ils appartiennent à la même cellule. Le plongement des différents types de cellule sont exploités lors de la mise en place de systèmes d'animation et de simulation physique.

2.2 Modèles multirésolutions

Les modèles topologiques présentés jusqu'ici représentent un seul maillage. Il est bien sûr possible de modifier ce maillage en fonction des besoins, mais cela se fait en général sans conserver l'historique des opérations et donc en perdant les informations concernant le maillage modifié. Pour remédier à cela, différentes approches ont été proposées comme les maillages progressifs et les modèles multirésolutions qui permettent de modéliser une série de maillages plus ou moins structurée. En général, ces maillages sont utilisés pour représenter différentes discrétisations du même objet, voir par exemple la figure 2.9.

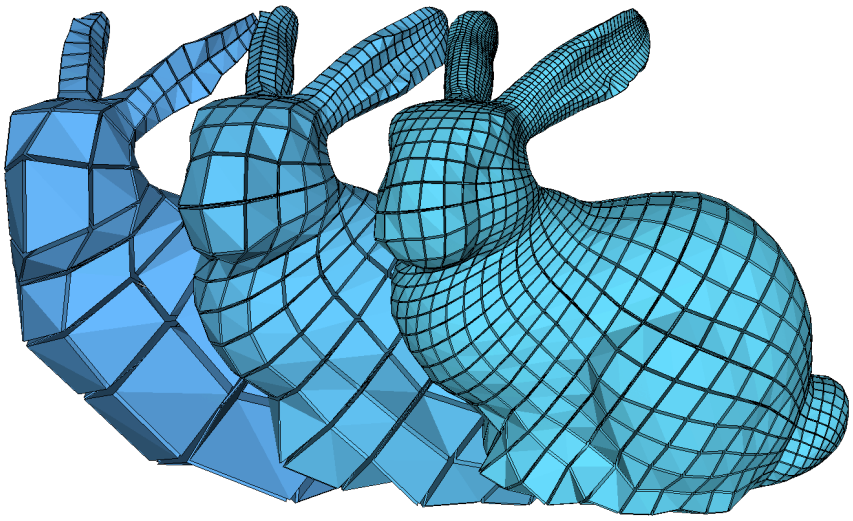

FIGURE 2.9 – Exemple de hiérarchie multirésolution, ici trois niveaux de résolution, le plus grossier à gauche, l'intermédiaire au milieu et le plus fin à droite.

Cette caractéristique ouvre la voie à de nombreuses applications habituellement réparties en deux grandes catégories : celles qui visent à accélérer des traitements et celles qui exploitent la multirésolution en tant que telle pour proposer d'autres outils de modélisation. Dans la première catégorie, on retrouve, par exemple, le rendu d'objets en fonction du point de vue ou les algorithmes de détection de collisions. Dans la deuxième,

on trouve les travaux sur les surfaces ou volumes de subdivision ou encore des outils de filtrage sur des maillages, exploitant la multirésolution pour définir des outils de traitement du signal adaptés à la géométrie.

En général, on considère deux approches pour la création de modèles multirésolution : l'approche *fin vers grossier* qui part d'un maillage fin et le simplifie progressivement pour obtenir les versions plus grossières, et l'approche *grossier vers fin* qui part d'un maillage grossier et le subdivise progressivement pour obtenir une hiérarchie de maillages plus détaillés.

L'objectif d'un modèle topologique multirésolution est donc de pouvoir accéder aux différents niveaux de discrétisation ou de détail de l'objet modélisé, indépendamment l'un de l'autre, et de permettre des accès concurrents à la topologie de chacun des niveaux.

Une solution simple pour modéliser cette série de maillages consisterait à utiliser un ensemble de maillages monorésolutions et donc à représenter chaque résolution indépendamment. La limite d'une approche aussi basique est qu'il n'existerait justement aucun lien entre les différentes résolutions. Or en général, par construction, ces résolutions forment une hiérarchie qu'il peut être utile d'exploiter. Enrichir un ensemble de maillages modélisés de manière indépendante pour représenter de tels liens hiérarchiques serait coûteux et peu productif. L'objectif que nous poursuivons au travers l'usage de modèles multirésolutions est ainsi d'encoder toutes les versions de l'objet et de les lier entre elles tout en limitant au maximum la redondance d'information. Nous souhaitons ainsi conserver un modèle topologique compact et efficace, tout conservant la possibilité de passer d'un niveau de détail à un autre, en fonction des besoins.

Dans les pages qui suivent, nous présentons les modèles multirésolutions les plus représentatifs de l'état de l'art, pour montrer leurs atouts et limitations, avant de présenter les cartes combinatoires multirésolution.

2.2.1 Maillages progressifs

Les maillages progressifs proposés par H. Hoppe [Hop96] sont une première approche multirésolution qui a été très exploité pour le rendu temps réel de maillages de grandes tailles. Ils sont basés sur une approche fin vers grossier et utilisent l'opérateur de contraction d'arêtes, qui consiste à supprimer une arête d'un maillage en fusionnant ses deux sommets, pour construire une hiérarchie.

Pour construire un maillage progressif, on part d'un maillage triangulaire M sur lequel est appliquée une série de n contractions d'arêtes disjointes. À chaque contraction, on construit une nouvelle version plus grossière du maillage initial. A la fin, on dispose donc de $n+1$ niveaux de détail de plus en plus grossier jusqu'à M_0 le maillage le plus grossier.

L'opération inverse de la contraction d'arêtes est l'éclatement de sommet. Il est possible, à partir de M et de la liste de contraction d'arêtes, de définir la liste des éclatements de sommet permettant de retrouver M à partir de M_0, en passant par tous les maillages intermédiaires. La figure 2.10 illustre les opérations de contraction d'arête et d'éclatement de sommet en dimension 2 et 3.

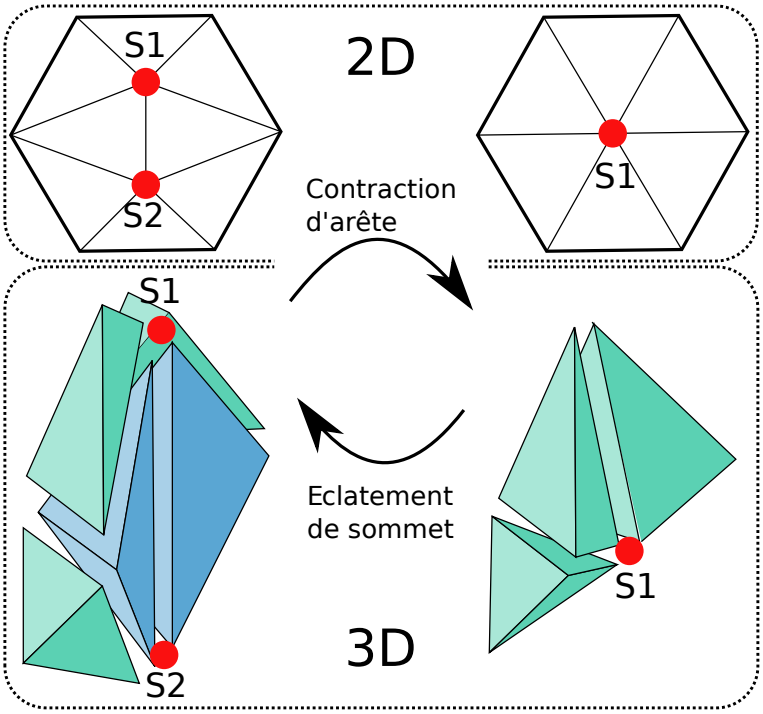

FIGURE 2.10 – Contraction d'arête et éclatement de sommet.

La première version des maillages progressifs ne permettait pas choisir les contractions d'arêtes à appliquer. L'ordre des opérations est figé et donc pour passer d'un niveau de résolution à un autre, il faut appliquer toutes les contractions (ou éclatements) séparant les deux niveaux.

Une amélioration a été proposée pour permettre une application adaptative des contractions ou éclatements. Cette approche consiste à représenter la hiérarchie d'éclatement de sommets comme une forêt d'arbres binaires où chaque sommet de M_0 est une racine d'un arbre. En maintenant un front de sommets actifs, il est possible d'appliquer une partie des éclatements de sommets en descendant plus ou moins profondément dans chacun des arbres. Enfin, une extension en dimension 3 a été proposée pour définir des maillages progressifs tétraédriques [SG98]. Ce modèle permet d'encoder une hiérarchie multirésolution de maillages tétraédriques en utilisant une contraction d'arête volumique, voir figure 2.10.

2.2.2 Quadtrees et Octrees

Une autre méthode très répandue pour représenter des hiérarchies multirésolutions est d'exploiter une structure à base d'arbres. C'est le cas des quadtrees pour les maillages surfaciques et des octrees pour les maillages volumiques. La filiation dans les arbres est ici liée à l'opération de subdivision de cellule.

Un quadtree représente la hiérarchie des résolutions d'un maillage avec un arbre quaternaire, ou plus précisément par une forêt d'arbres quaternaires dont les racines sont placées sur les faces du maillage initial. Dans ces arbres, chaque nœud lie une face mère à quatre sous-faces filles comme illustré dans la figure 2.11. L'utilisation d'un opérateur de subdivision unique limite l'usage des quatrees à l'approche grossier vers fin.

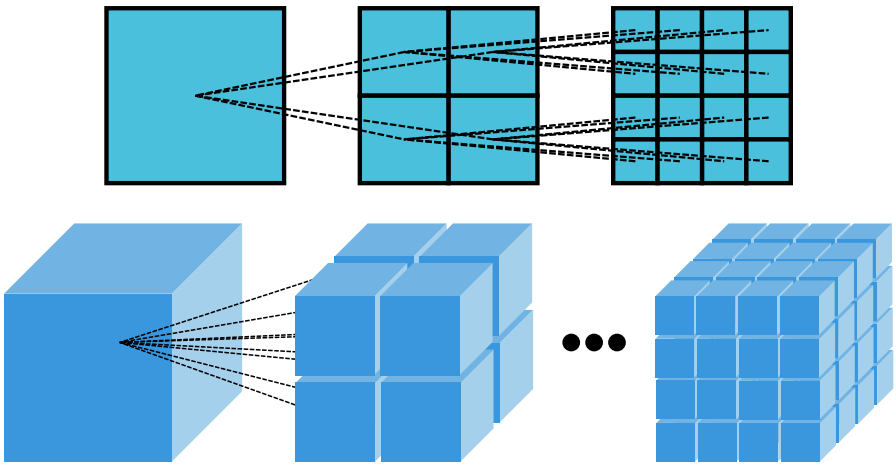

FIGURE 2.11 – Illustration du raffinement d'un quadtree (haut) et d'un octree (bas)

Une des principales limitations des quatrees est liée au fait que toutes les cellules ne sont pas représentées explicitement. La subdivision d'une face, sans subdivision des faces adjacentes, introduit des sommets sur les arêtes incidentes. Ces sommets n'existent pas dans les faces adjacentes ce qui provoque l'apparition de *trous topologiques*. Un usage adaptatif des quatrees génère donc des maillages *non-conformes*.

Ce problème de non-conformité peut être résolu en regroupant entre elles les opérations topologiques aboutissant à de telles non-conformités, mais une telle contrainte réduit fortement le nombre de maillages représentables. Une seconde solution est d'utiliser des quadtrees restreints qui autorisent au maximum un niveau d'écart entre deux faces adjacentes et ajoutent des faces *virtuelles* là où des trous topologiques apparaissent, pour les combler. On parle alors de *jonctions en T*. Cette méthode est illustrée dans la figure 2.12 pour les octrees.

La seconde limitation est due au fait que dans un quadtree les relations d'adjacence ne sont pas non plus explicitement représentées. La régularité du raffinement et la simplicité de la structure d'arbre permet de retrouver ces relations au moyen d'un parcours de l'arbre, mais leur reconstruction est alors résolue dans un temps logarithmique du nombre de faces présentes dans le quadtree. Ce temps de calcul peut devenir rapidement prohibitif pour des applications réelles.

De la même manière, pour des maillages volumiques, des arbres octaux ou octrees sont utilisés. Comme pour les quadtrees, chaque volume du maillage initial est subdivisé en 8 sous-volumes. Les octrees souffrent des mêmes limitations que les quadtrees.

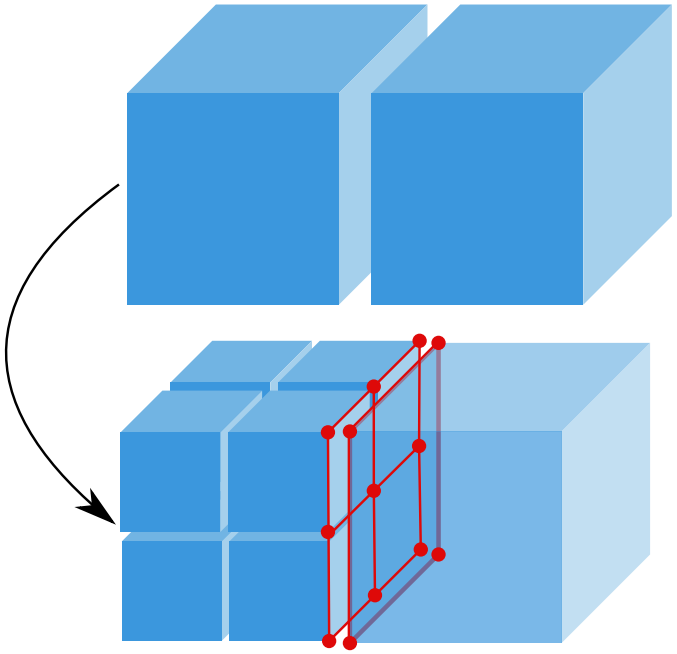

FIGURE 2.12 – Jonction en T (en rouge) ou trou topologique dans un octree entre deux volumes subdivisés à des niveaux différents.

2.3 Cartes combinatoires multirésolutions

2.3.1 Hypercartes multirésolutions

Le principe des modèles topologiques multirésolution est de permettre la représentation d'objets à différents niveaux de résolutions et de donner accès aux informations topologiques à chaque niveau. Nous transposons ce principe aux hypercartes avant de le décliner. Le principe est de partir d'une hypercarte représentant une version grossière d'un objet, puis de lui ajouter des brins, de manière itérative, en vue de représenter une succession de niveaux de résolution, jusqu'à arriver à la version la plus détaillée de l'objet. Une hypercarte classique est définie par un unique ensemble de brins. Dans une hypercarte multirésolution, plusieurs versions d'un même objet sont représentées, imbriquées les unes dans les autres.

Représenter une telle hiérarchie de façon cohérente nécessite, à chaque résolution, de réutiliser les brins des résolutions précédentes en y ajoutant de nouveaux brins. D'un point de vue formel, cela forme une suite imbriquée d'hypercartes pour laquelle $B^0 \subset B^1 \subset \ldots \subset B^{n-1} \subset B^n$, où B^0 est l'ensemble des brins de l'hypercarte la plus grossière et B^n est l'ensemble des brins de l'hypercarte la plus détaillée. Un brin ajouté à un niveau k est présent dans l'ensemble des niveaux supérieurs à k. Pour représenter sans redondance cette hiérarchie, nous définissons N^k l'ensemble des brins ajoutés au niveau k, c'est-à-dire $B^k \setminus B^{k-1}$. Ainsi, B^k est défini par : $B^k = \bigcup_{0 \leq i \leq k} N^i$.

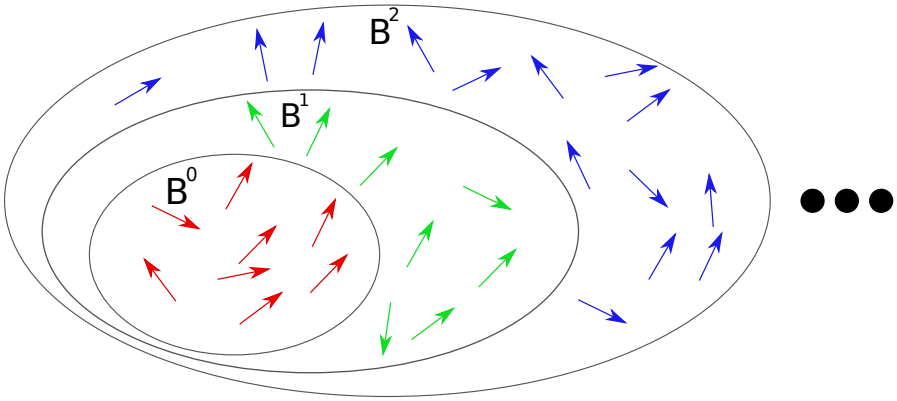

FIGURE 2.13 – Ensembles imbriqués de brins et leurs niveaux d'insertion : N^0 (rouge), N^1 (vert) et N^2 (bleu)

L'étape suivante consiste à représenter les relations entre les brins des B^k, aux différentes résolutions. Dans une hypercarte multirésolution, il est souhaitable que les brins d'une résolution puissent être liés aux brins des résolutions inférieures, le contraire impliquant la formation de composantes connexes à chaque nouvelle résolution. En complément, les relations établies à un niveau B^k doivent pouvoir être modifiées dans les niveaux B^l avec $l > k$ ou rester inchangées. Enfin, le modèle doit permettre de parcourir chaque niveau de résolution de manière indépendante.

Cela nous amène à définir, pour chaque niveau de résolution k, un ensemble de relations ϕ_i^k liant les brins par ϕ_i au niveau k. Par exemple ϕ_1^0 relie entre eux les brins de l'ensemble B^0 et ϕ_1^1 relie des brins de B^1. Pour un brin b de B^0, on peut avoir $\phi_1^1(b) = \phi_1^0(b)$ indiquant que la relation reste inchangée entre les niveaux 0 et 1. Pour un brin c de B^1 n'existant pas dans B^0, on peut avoir $\phi_1^1(c) = b$ ce qui permet de lier des cellules de niveau 0 et 1. A ce stade, aucune restriction n'est imposée sur ces relations ce qui nous amène à la définition suivante :

Définition 7 *Une hypercarte multirésolution de dimension n est un ensemble d'hypercartes imbriquées, définie par un (n+1)-uplet :*

$$H = (\{B^k\}_{k\geq 0}, \{\phi_0^k\}_{k\geq 0}, ..., \{\phi_{n-1}^k\}_{k\geq 0})$$

tel que :
— $H^k = (B^k, \phi_0^k, ..., \phi_n^k)$ *est une hypercarte*
— $\forall i, j : 0 \leq i \leq j \leq n,\ B^i \cap B^j = B^i$

La première condition implique que les ϕ_i^k sont des permutations de B^k dans B^k. La seconde condition indique que les B^k sont imbriqués les uns dans les autres et forment une hiérarchie.

Il faut noter que jusqu'ici aucune contrainte n'a été imposée quant à la cohérence topologique ou géométrique des niveaux de résolution que représentent les H^k entre eux.

À ce stade, deux résolutions peuvent donc représenter des objets totalement différents aussi bien d'un point de vue topologique que géométrique.

2.3.2 Cartes combinatoires multirésolutions

Nous pouvons maintenant définir une carte combinatoire multirésolution aussi bien primale que duale à partir de la définition des hypercartes multirésolution :

Définition 8 *Une carte combinatoire multirésolution, dite primale, de dimension n est un (n+1)-uplet* $C = (\{B^k\}_{k\geq 0}, \{\alpha_0^k\}_{k\geq 0}, ..., \{\alpha_{n-1}^k\}_{k\geq 0})$ *tel que :*
— $C^k = (B^k, \alpha_0^k, ..., \alpha_{n-1}^k)$ *est une carte combinatoire primale*
— $\forall i, j : 0 \leq i \leq j \leq n,\ B^i \cap B^j = B^i$

Définition 9 *Une carte combinatoire multirésolution, dite duale, de dimension n est un (n+1)-uplet* $C = (\{B^k\}_{k\geq 0}, \{\phi_1^k\}_{k\geq 0}, ..., \{\phi_n^k\}_{k\geq 0})$ *tel que :*
— $C^k = (B^k, \phi_1^k, ..., \phi_n^k)$ *est une carte combinatoire duale*
— $\forall i, j : 0 \leq i \leq j \leq n,\ B^i \cap B^j = B^i$

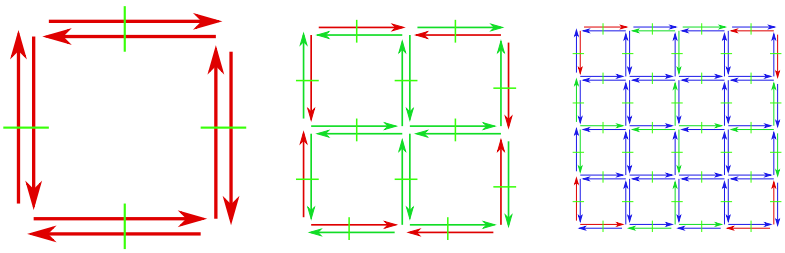

FIGURE 2.14 – Exemple de carte combinatoire multiresolution en dimension 2, de gauche à droite : niveau 0,1 et 2. La couleur des brins correspond à leurs niveaux d'insertion selon les mêmes couleurs que la Figure 2.13.

La figure 2.14 illustre la notion de carte multirésolution et la hiérarchie formée par les ensembles de brins. Les deux définitions indiquent que chaque niveau de résolution est une carte ce qui garantit qu'il représente une quasi-variété orientable et fournit le moyen de parcourir les cellules de différentes résolutions. Par exemple, dans une 3-carte duale, les brins d'un sommet de résolution k sont atteints par l'orbite $< \phi_1^k \circ \phi_2^k, \phi_1^k \circ \phi_3^k >$, les brins d'une arête par l'orbite $< \phi_2^k, \phi_3^k >$, les brins d'une face par l'orbite $< \phi_1^k, \phi_3^k >$ et les brins d'un volume par l'orbite $< \phi_1^k, \phi_2^k >$.

Comme pour les hypercartes, aucune contrainte n'est imposée entre les différents niveaux de résolution. Ces contraintes sont mises en place au niveau des opérations de manipulation définies ci-après.

2.3.3 Manipulations topologiques

Après avoir défini formellement les cartes combinatoires multirésolutions, nous montrons maintenant comment construire une hiérarchie au travers d'opérateurs de manipulation topologique simples.

Rappelons que notre objectif est de construire une hiérarchie de cartes de manière que chaque niveau de résolution représente le même objet avec des détails supplémentaires, nous appellerons un détail de niveau k chaque cellule ajoutée au niveau de résolution k. Ces opérations doivent de plus respecter les contraintes des cartes combinatoires.

Un invariant topologique classique permettant de décrire une variété de dimension n est la caractéristique d'Euler-Poincaré, définie comme la somme alternée du nombre de cellules de la décomposition cellulaire de l'objet : $\chi = \sum_{i=0}^{n}((-1)^i * c_i)$. Par exemple, pour la 3-carte de la Figure 2.2, on a : 12 sommets, 20 arêtes, 11 faces et 3 volumes (la 3-carte étant sans bord, elle possède un volume englobant), sa caractéristique d'Euler-Poincaré est donc : $\chi = 12 - 20 + 11 - 3 = 0$. Pour conserver la nature topologique des objets manipulés, nous utiliserons ici des opérations topologiques conservant la caractéristique d'Euler-Poincaré, bien que cela ne soit pas une obligation dans le cas général.

Pour conserver la caractéristique d'Euler-Poincaré, il suffit de définir des opérations telles que l'ajout d'une k-cellule soit accompagné de l'ajout d'une $(k+1)$-cellule pour $k < n$. Par exemple, l'ajout d'un sommet doit être accompagné de l'ajout d'une arête ou l'ajout d'une face doit être accompagné de l'ajout d'un volume. Cette observation nous permettra de définir trois opérations topologiques élémentaires consistant à découper une $(k+1)$-cellule en deux $(k+1)$-cellule par l'ajout d'une k-cellule. Nous commençons par définir ces opérations dans un cadre monorésolution, pour les exploiter ensuite pour permettre la création d'une hiérarchie de cartes.

Découpe d'arête

L'opération de découpe d'arête consiste à remplacer une arête constituée du couple de sommet $\{S_1; S_2\}$ par deux arêtes $\{S_1; S_k\}$ et $\{S_k; S_2\}$ séparées par un sommet central commun S_k en réutilisant les sommets de l'arête précédente. Nous détaillons cette opération en 2D, puis en 3D.

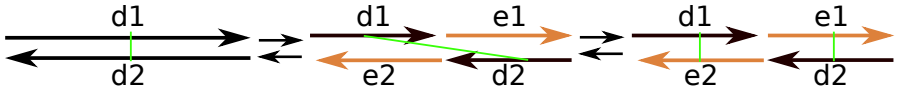

FIGURE 2.15 – Découpe d'arête dans une 2-carte, les nouveaux brins sont en orange.

Dans une 2-carte, une arête est constituée de deux brins d_1 et d_2. La découpe revient à insérer entre eux deux brins e_1 et e_2 et à modifier les relations de la façon suivante : $\phi_1(e_1) := \phi_1(d_1)$, $\phi_1(e_2) := \phi_1(d_2)$, $\phi_1(d_2) := e_2$, $\phi_1(d_1) := e_1$, $\phi_2(d_1) := e_2$, $\phi_2(d_2) := e_1$, $\phi_2(e_1) := d_2$ et $\phi_2(e_2) := d_1$. Cette opération est illustrée dans la Figure 2.15.

Pour une 3-carte l'opération est similaire et consiste à couper l'arête sur chacun des

volumes incidents, puis à mettre à jour les relations ϕ_3 pour intégrer les nouveaux brins. Cette opération est illustrée sur la Figure 2.16.

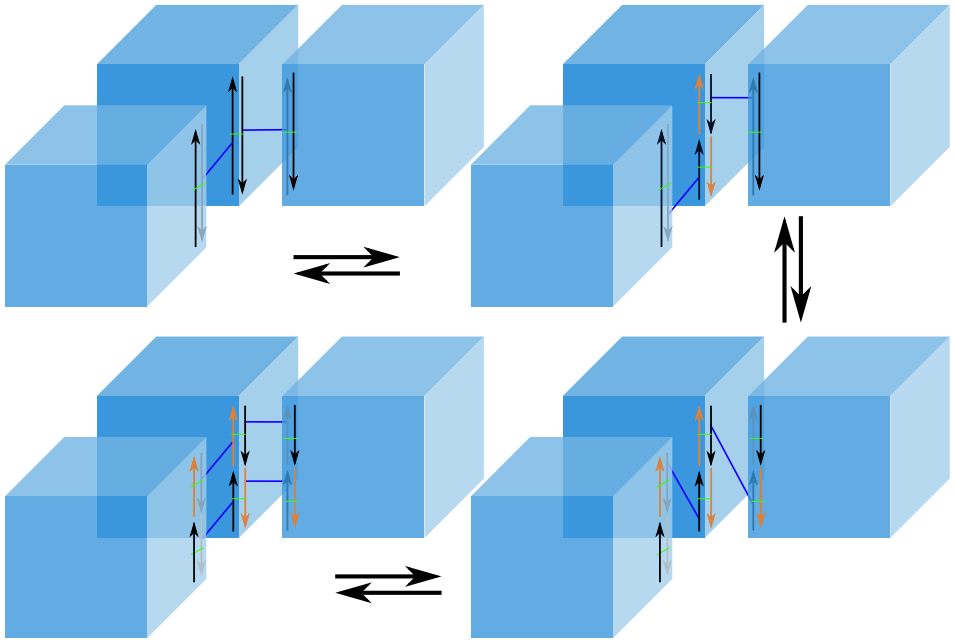

FIGURE 2.16 – Découpe d'arête dans une 3-carte multirésolution. Pour un besoin de visibilité seuls les brins de l'arête découpée sont représentés. Les brins non représentés ne sont pas impactés par la découpe.

Découpe de face

La découpe d'une face dans une 2-carte est réalisée en reliant deux sommets non consécutifs par une arête constituée de deux nouveaux brins reliés par la relation ϕ_2 comme illustré dans la Figure 2.17.

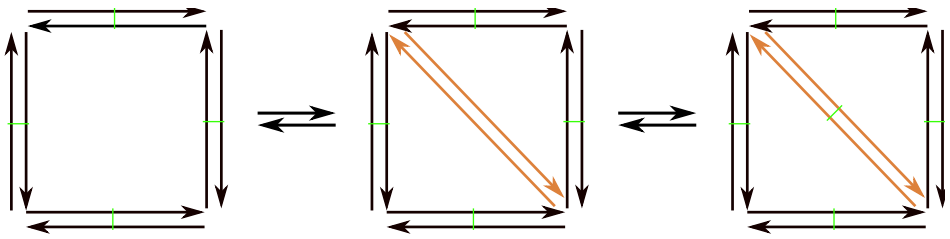

FIGURE 2.17 – Découpe d'une face d'une 2-carte, les nouveaux brins sont en orange.

Dans une 3-carte, une 3-face est formée de deux 2-faces, cousues entrent-elles par la

relation ϕ_3. Sa découpe consiste à découper les deux 2-faces, comme dans une 2-carte, puis à relier les brins nouvellement créés par ϕ_3 comme illustré sur la Figure 2.18.

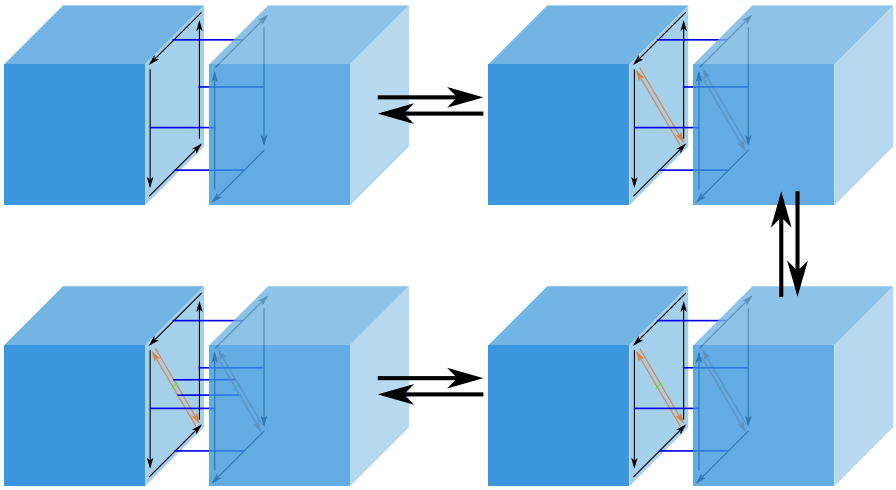

FIGURE 2.18 – Découpe d'une face dans une 3-carte, pour un besoin de lisibilité seuls les brins de la face découpée sont représentés.

Découpe de volume

La découpe d'un volume consiste à insérer une face le long d'un chemin d'arêtes. Les deux nouveaux volumes sont liés par ϕ_3 le long de cette face commune. En pratique la découpe est réalisée en créant une 3-face de degré égal au nombre d'arêtes dans le chemin choisi, puis en cousant par ϕ_2 chaque arête de cette face à un brin du chemin choisi, comme illustré dans la Figure 2.19.

Notion de sous-cellules

La découpe des différentes cellules entraîne la création de sous-cellules. Par exemple, les deux arêtes ou les quatre faces créées après une découpe d'arête ou de face seront appelées des sous-cellules de la cellule découpée. Ce concept de sous-cellules sera essentiel dans les parties suivantes.

Schéma de subdivision et multirésolution

Après avoir examiné les opérateurs topologiques élémentaires, nous les combinons pour implémenter le schéma de subdivision *Butterfly* [DSL93] qui nous permet de créer une structure hiérarchique multirésolution. Ce schéma agit, par itération successive, sur la topologie et le plongement d'un objet pour obtenir une forme de plus en plus lisse. Nous illustrons la subdivision topologique correspondant dans le cas spécifique de la subdivision d'un hexaèdre.

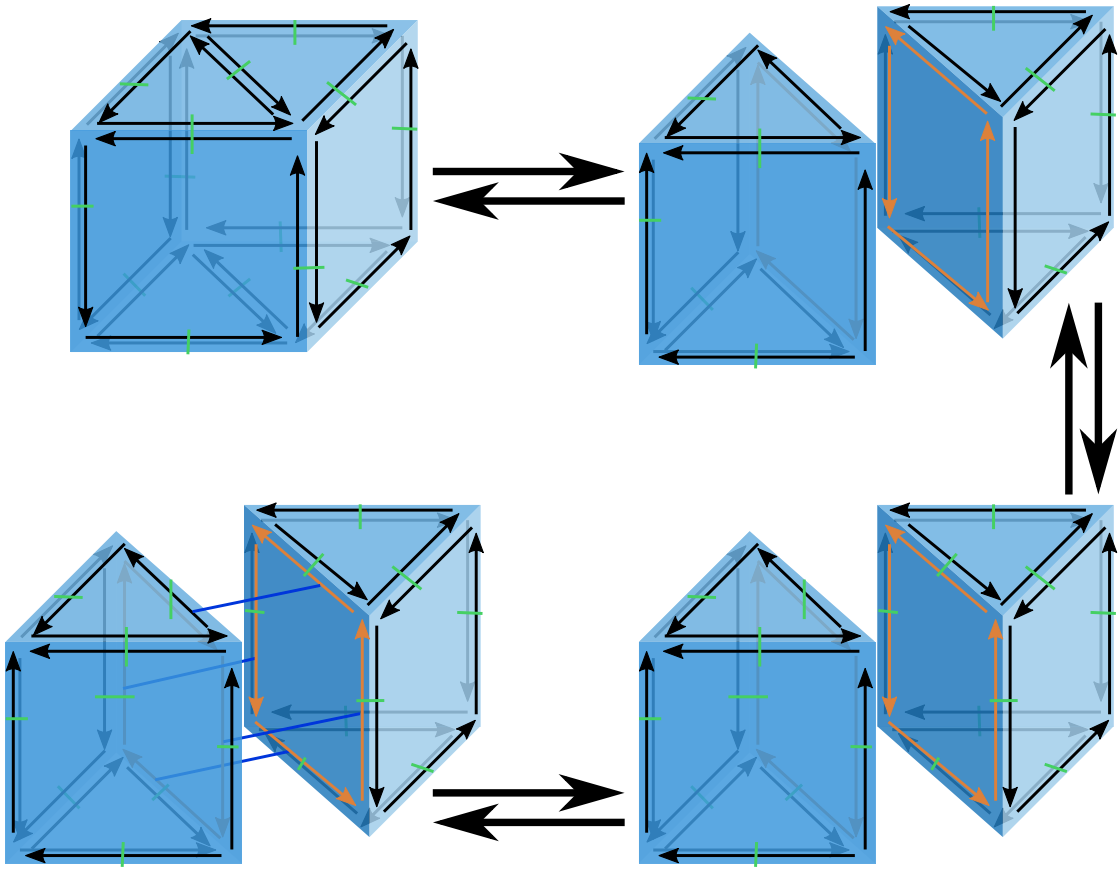

FIGURE 2.19 – Découpe d'un volume dans une 3-carte.

Celle-ci se décompose en trois étapes : la découpe des arêtes en deux, la découpe des faces en quatre en utilisant les sommets insérés dans les arêtes, puis la découpe des volumes en huit. La Figure 2.20 illustre cet algorithme. Il est important de souligner que les faces intérieures insérées lors des découpes de volumes sont également découpées pour permettre les découpes suivantes. Une fois un niveau créé, son plongement peut être modifié pour appliquer la partie géométrique du masque de subdivision.

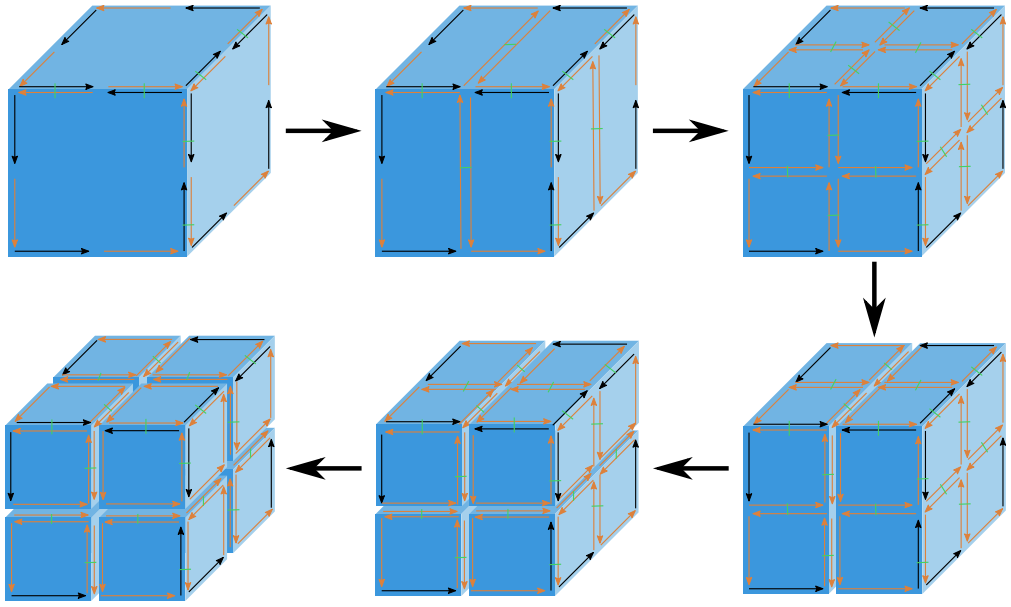

FIGURE 2.20 – Découpe topologique d'un hexaèdre en huit en appliquant les opérateur de découpe de base. Pour un besoin de lisibilité seul les brins visibles et leurs relations sont représentés.

Pour créer une hiérarchie multirésolution, nous partons d'une carte contenant un seul niveau 0, puis nous ajoutons des niveaux. Dans une carte à k niveau, pour ajouter un niveau, il suffit de dupliquer le niveau k au niveau $k+1$ puis d'appliquer le schéma de subdivision au niveau $k+1$. Le schéma peut être appliqué de manière adaptative, c'est-à-dire partiellement. Ce principe est illustré dans la Figure 2.21.

Niveau d'une cellule

Un concept essentiel lorsque l'on construit une hiérarchie est le niveau d'une cellule. Il représente intuitivement le niveau auquel la cellule apparaît. Une approche naïve consisterait à rechercher, au sein de la cellule, le brin le plus récent, c'est-à-dire le brin de niveau le plus élevé. Cette approche serait correcte dans le cas de subdivisions uniformes, mais ne l'est pas dans le contexte adaptatif que nous avons adopté.

En effet, le bord d'une cellule peut être subdivisé sans que celle-ci le soit. Cette configuration survient, par exemple, lorsqu'on subdivise un volume sans subdiviser un

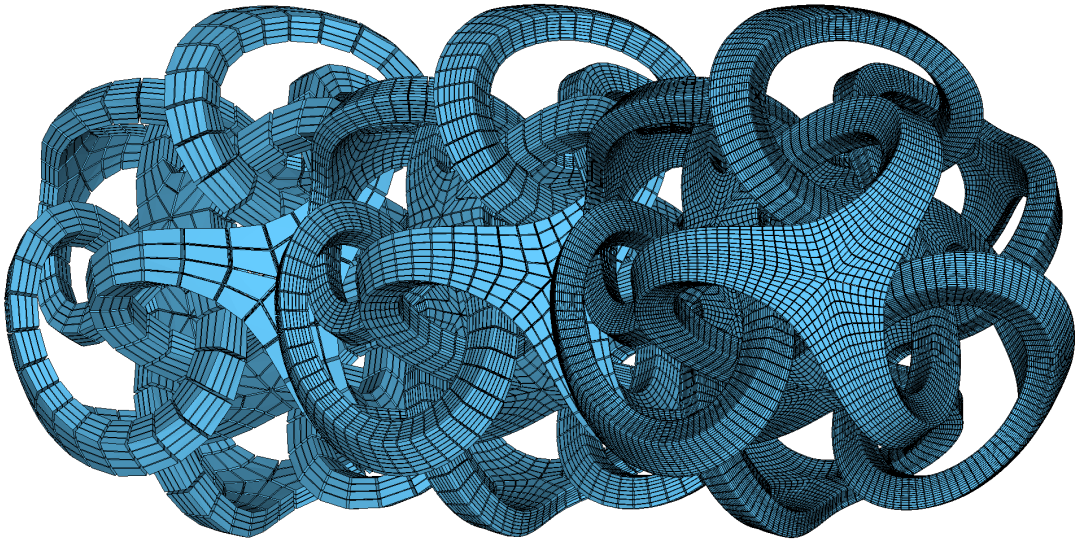

FIGURE 2.21 – Application de schéma *Butterfly* sur un objet complexe, pour générer deux niveaux de résolution.

des volumes adjacents. Dans ce cas, la face incidente aux deux volumes est subdivisée. Nous proposons une définition plus précise du niveau d'une cellule :

Définition 10 *On note* $cell_k^j(\mathcal{M}, d)$ *l'ensemble des brins de la k-cellule de d à la résolution j de la carte multirésolution* $\mathcal{M}$*, c'est-à-dire les brins de l'orbite correspondant à une k-cellule, observé au niveau j.*

A la résolution j, le niveau de la k-cellule associée à d pour $k \geq 1$ *est le plus petit niveau* $i \geq 0$ *tel que* $cell_k^i(\mathcal{M}, d) \subset cell_k^j(\mathcal{M}, d)$*. On note* $niveau_k(\mathcal{M}, d)$ *ce niveau ou* $niveau_k(d)$ *dans les situations où la carte n'a pas besoin d'être précisée.*

Cette notion de niveau de cellule est illustrée dans la Figure 2.22. Notons que les opérations de subdivision utilisées ici ne modifient jamais les orbites sommets. En effet, les opérations de subdivision d'arête, de face ou de volume ajoutent des nouveaux sommets, mais n'ajoutent aucun brin à l'orbite d'un sommet existant. Ainsi le niveau d'un sommet est simplement le niveau d'un de ses brins.

2.3.4 Représentation des cartes combinatoires multirésolutions

Les cartes combinatoires multirésolution ont déjà été étudiées dans deux travaux de thèse différents, pour représenter des surfaces de subdivision dans la thèse de P.Kraemer [Kra08] et pour représenter des volumes de subdivision dans la thèse de L.Untereiner [Unt13]. Nous présentons ci-après les structures de données et méthodes utilisées pour implanter efficacement les cartes multirésolution.

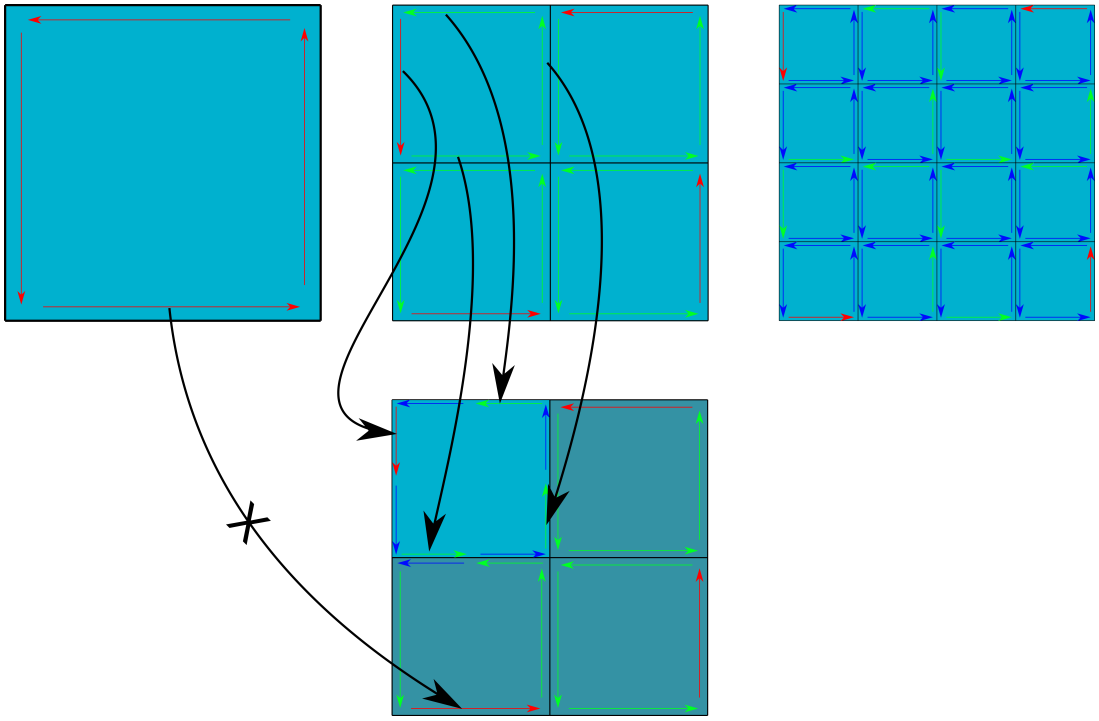

FIGURE 2.22 – Recherche du niveau d'une face. En haut : une face de niveau 0 et ses deux subdivisions. En bas : une face adaptative. Cette face possède tous les brins d'une face de niveau 1 (correspondance illustrée par les flèches noires), mais pas tous ceux de niveau 0 (comme indiqué par la flèche barée). Elle est donc de niveau 1.

Cartes combinatoires multirésolutions explicites

Une première façon de représenter une carte combinatoire multirésolution est la représentation dite explicite. Chaque niveau de résolution est représenté par une carte combinatoire différente ce qui est exactement la définition donnée précédemment.

En pratique, au lieu de stocker n ensembles de brins pour représenter les n niveaux de résolution, une seule collection d'ensembles de brins est conservée en mémoire. Chaque brin est associé à un entier indiquant son niveau d'insertion dans la hiérarchie multirésolution, c'est-à-dire à l'ensemble N^k auquel il appartient. Un ensemble de n relations ϕ_i est également maintenu pour exprimer l'ensemble des relations ϕ_i^k.

Le principal avantage de cette structure réside dans la similitude théorique entre le parcours topologique de chaque niveau de résolution et celui d'une carte monorésolution traditionnelle. Toutefois, dans la pratique, l'ajout d'une redirection mémoire pour accéder à la relation ϕ_i^k appropriée diminue les performances par rapport à un parcours de carte monorésolution classique. Il convient néanmoins de souligner l'économie de mémoire résultant de la réutilisation des ensembles de brins des niveaux inférieurs lors de l'introduction de nouveaux niveaux de résolution.

Cartes combinatoires multirésolutions implicites

La représentation explicite a l'avantage de représenter exactement le modèle théorique des cartes multirésolution, mais implique un stockage potentiellement important de données. Nous présentons maintenant une représentation plus compacte des cartes multirésolutions exploitant le fait que la hiérarchie est obtenue par un raffinement régulier. Il s'agit du modèle CPH pour *Compact Primal Hierarchy* introduit dans [Unt+15].

L'idée de la représentation implicite présentée dans le modèle CPH est de ne stocker en mémoire que la résolution la plus fine $M^k = \{D^k, \phi_1^k, ..., \phi_n^k\}$ et de reconstruire les cartes des niveaux inférieurs à l'aide d'algorithmes exploitant la connaissance du schéma de subdivision utilisé. Il nous faut donc d'une part reconstruire l'ensemble des brins D^i pour chaque niveau de résolution i ainsi que l'ensemble des relations $\phi_0^i, ..., \phi_n^i$, pour retrouver à la volée et implicitement les $M^i = \{D^i, \phi_1^i, ..., \phi_n^i\}$.

Pour reconstruire l'ensemble des brins, chaque brin dispose simplement d'un entier représentant son niveau d'insertion. L'ensemble des brins D^i est ainsi constitué de l'ensemble des brins ayant un niveau d'insertion inférieur ou égal à i. Les brins peuvent être triés par niveau d'insertion pour parcourir très rapidement ces ensembles.

La reconstruction des relations entre les brins est moins triviale. La première relation à reconstruire est la relation ϕ_1^i, les autres relations pouvant être retrouvées à partir de cette relation. L'idée est de trouver un chemin au sein des brins de D^k utilisant les relations de M^k pour retrouver le brin correspondant à la relation ϕ_1^0 de la carte M^0. Ce chemin est illustré dans la Figure 2.23.

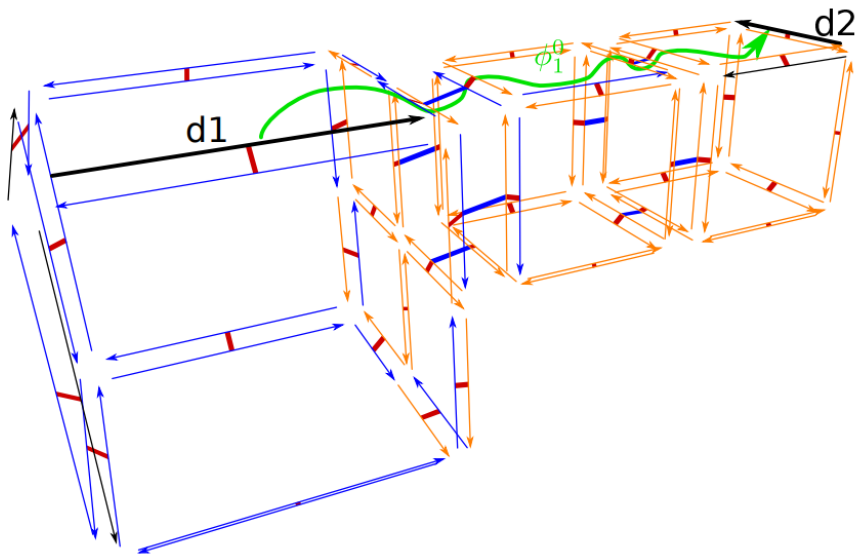

FIGURE 2.23 – Illustration du chemin parcouru pour reconstruire $\phi_1^0(d1)$ en utilisant les relations de la résolution k on remarque le besoin de pouvoir *sauter* au-dessus d'arêtes.(issu de [Unt+15])

On remarque que ce chemin consiste à *sauter* par-dessus les brins ajoutés lors des différentes subdivisions. Pour cela nous avons besoin de savoir quand le chemin suivi revient sur l'arête d'origine. De même pour les faces il faut savoir quand le chemin revient sur la face d'origine. Cette information est retrouvée à l'aide d'un ingénieux système d'étiquettes.

L'algorithme 1, présenté ci-après, permet de reconstruire ϕ_1^i à toute résolution i. Il consiste à sauter les brins des sommets intermédiaires jusqu'à retomber sur une arête (ou une face) portant la même étiquette que l'arête (ou la face) de départ. À partir de ϕ_1^i, on peut retrouver ϕ_2^i qui est défini comme $\phi_2^i(d) = \phi_2^k(\phi_{-1}^k(\phi_1^i(d)))$ et ϕ_3^i qui est défini par $\phi_3^i(d) = \phi_3^k(\phi_{-1}^k(\phi_1^i(d)))$.

Algorithme 1 $\phi_1^i(d)$

```
1: finished ← False
2: l = edgeLabel(d)
3: it = d
4: repeat
5:     it = φ1^k(it)
6:     si insertionLevel(it) ≤ i alors
7:         finished = True
8:     sinon
9:         tant que edgeLabel(it) ≠ l faire
10:            l2 = faceLabel(it)
11:            it2 = φ2^k(it)
12:            tant que faceLabel(it2) ≠ l2 faire
13:                it2 = φ2^k(φ3^k(it2))
14:            fin tant que
15:            it = φ1^k(it2)
16:        fin tant que
17:    fin si
18: until finished
19: retourne it
```

Les étiquettes sont placées sur les faces et sur les arêtes comme illustré dans la Figure 2.24 de manière que des arêtes incidentes à un même sommet ne portent pas la même étiquette. La combinatoire très simple du schéma de subdivision permet de n'utiliser que 3 étiquettes, indépendamment du nombre de brins, d'arêtes ou de faces présents dans les différentes résolutions.

La représentation implicite permet d'implanter les cartes combinatoires multirésolutions de manière compacte, au prix de performances dégradées pour le parcours des niveaux de résolution les plus bas. En effet, le coût de l'algorithme de reconstruction des ϕ_1^i augmente avec la différence $k - i$, où k est le niveau maximal ou le plus fin de la carte. Le nombre de brins à sauter dépend du nombre de subdivisions existant entre le niveau i parcouru et le niveau k. Néanmoins, notons que les niveaux de résolution le

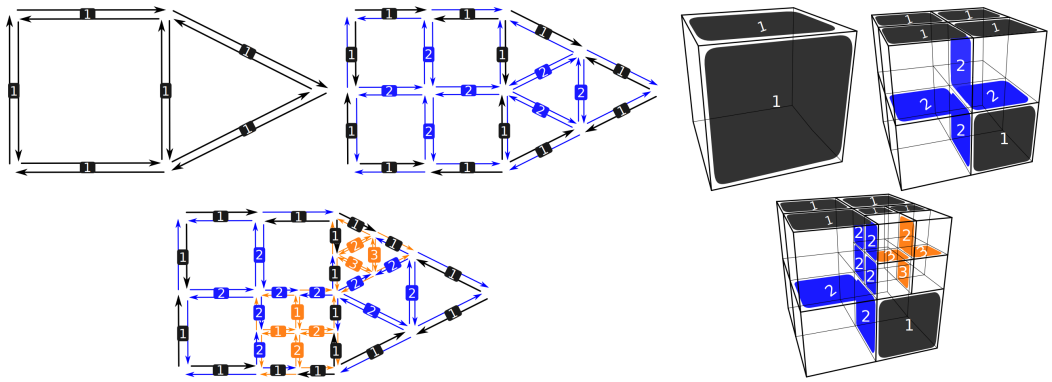

FIGURE 2.24 – Système d'étiquettes sur les arêtes (gauche) et les faces (droite). (issu de [Unt+15])

plus petits sont constitués d'un nombre de brins beaucoup plus faible que les niveaux élevés et donc que la dégradation des performances pour ces niveaux peut être considéré comme négligeable dans la plupart des applications. Plus précisément, $7/8$ des brins sont de niveau k et pour ceux-ci ϕ_1^k est obtenu directement. Parmi les $1/8$ brins de niveau inférieur à k, $7/8$, soit $7/8^2$ brins, sont de niveau $k-1$ et pour eux ϕ_1^{k-1} est obtenu en sautant au plus 1 sommet. De même pour les $7/8^3$ brins de niveau $k-2$, ϕ_1^{k-2} est obtenu en sautant au plus 2^2-1 sommets et ainsi de suite jusqu'aux $7/8^{(k+1)}$ brins de niveau 0 pour lesquels ϕ_1^0 est obtenu en sautant au plus 2^k-1 sommets.

Chapitre 3

Vues multirésolutions adaptatives

3.1 Présentation

Nous avons vu dans les sections précédentes différentes méthodes pour représenter des maillages multirésolutions. Les cartes combinatoires multirésolutions sont compactes et très efficaces et permettent des parcours optimisés de l'ensemble de niveaux de résolution représentés. Leur principale limitation concerne le fait de ne pas pouvoir parcourir les niveaux de résolution de manière adaptative, c'est-à-dire en décidant localement du niveau des cellules parcourues. La Figure 3.1 illustre ce principe avec deux parcours adaptatifs d'un maillage multirésolution.

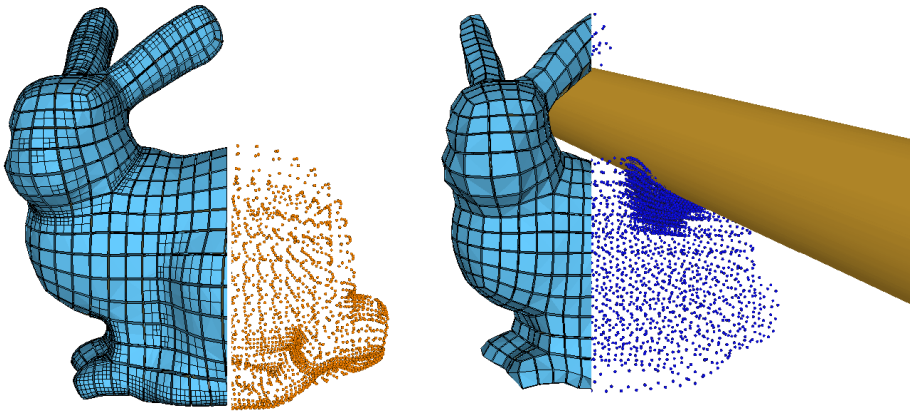

FIGURE 3.1 – Deux vues adaptatives du même lapin multirésolution sur lesquelles différents brins et cellules sont rendus visibles. À gauche : une adaptation de la surface limite en fonction de sa courbure pour le rendu. À droite : une adaptation de la densité de degrés de liberté dans une zone déformée.

Une approche naïve, consisterait à marquer les cellules *actives* que l'on souhaite parcourir, par exemple en étiquetant les sommets actifs à chaque niveau de résolution.

Cette approche nécessiterait le développement de parcours hiérarchiques en profondeur des cellules (similaires au parcours d'un octree), entraînant un surcoût de temps important car potentiellement toutes les cellules de tous les niveaux de résolution devraient être parcourus à chaque fois. Une première solution consisterait à maintenir un *front* de sommets actifs, comme cela a été proposé pour les maillages progressifs. Cependant, cette approche soulève d'autres difficultés telles que la complexité et le coût des mises à jour de ce front lors de changements topologiques.

De plus, nous voulons pouvoir réaliser des parcours adaptatifs en parallèle, pour des tâches telles que la simulation, le rendu ou la détection de collision. Cela nous a conduits à développer le concept de *vues* multirésolutions présenté ci-après.

Le principe général est de rendre visibles à une résolution i des sommets introduits à une résolution j plus élevée. Pour cela, nous attribuons une étiquette visibilité(d) à chaque brin d, indiquant le niveau à partir duquel il doit être visible. Notez que nous n'autorisons pas un brin à devenir invisible. Cela implique d'ajouter la contrainte d'intégrité suivante : visibilité(d) $\leq$ *niveau*(d) pour tout d.

Notre objectif est de définir, pour un niveau donné i, un maillage cohérent qui contient les sommets de $\mathcal{M}$ visibles au niveau i, comme illustré dans la figure 3.1. Cela nécessite de rendre les brins incidents à ces sommets visibles au niveau i, ainsi que leurs cellules environnantes (arêtes, faces ou volumes). Les nouvelles relations de voisinage de ces cellules doivent être cohérentes avec les relations des cellules plus grossières déjà présentes au niveau i. C'est le principal défi abordé dans les sections suivantes.

3.2 Définition

Une vue multirésolution adaptative $\mathcal{V}_i$ est une carte combinatoire construite à partir de l'ensemble des brins visibles à la résolution i :

Définition 11 *Une vue adaptative d'une carte multirésolution $\mathcal{M}$ de dimension n et de niveau maximal k est un $(n+2)$-uplet $\mathcal{V} = (\mathcal{M}, v, \{\psi_1^i\}_{0\leq i\leq k}, ..., \{\psi_n^i\}_{0\leq i\leq k})$ tel que :*

- *v est une fonction de $B^k \rightarrow [0,k]$ qui définit la visibilité d'un brin*
- *on note V^i l'ensemble $\{b, v(b) \leq i\}$ des brins visibles au niveau i avec $i \leq k$*
- *la visibilité v respecte la contrainte $B^i \subset V^i \subseteq B^k$ $\forall i \leq k$*
- *$\forall i \leq k,\ M^i = (V^i, \psi_1^i, ..., \psi_n^i)$ est une carte combinatoire (définition 4)*

Notre objectif est de définir les ψ_i^k d'une vue à partir des ϕ_i^k de la carte multirésolution sous-jacente afin que la topologie de la vue s'adapte de manière automatique et *à la volée* aux changements de visibilité opérés sur la carte.

Dans la suite du texte, nous simplifierons les notations en omettant les exposants sur les ensembles de brins et les relations entre les brins lorsque le niveau de résolution est défini sans ambiguïté. Dans ces cas, les explications seront considérées comme valables pour tous les niveaux de résolution. De plus, lorsque plusieurs relations ou ensembles de brins sont évoqués simultanément, ils seront considérés à priori comme étant au même niveau de résolution, ce qui facilitera la lisibilité des analyses et explications données.

Ci-après, nous présentons les propriétés attendues des vues qui vont nous servir à définir les contraintes sur ce modèle. Le concept de vue consiste à ajouter des détails à différents niveaux de résolution. Si aucun détail n'a été ajouté jusqu'au niveau i, c'est-à-dire si aucun brin de niveau supérieur à i n'a été rendu visible à un niveau inférieur à i, alors pour tout $j \leq i$ les relations ψ_k^j sont identiques aux ϕ_k^j, c'est-à-dire que la vue représente la même topologie que la carte sous-jacente jusqu'au niveau i.

Propriété 1 $\forall i \in [0,k]$, *si* $V^i = B^i$, *alors* $\forall j \in [0,n]$, $\psi_j^i = \phi_j^i$.

En complément, si tous les brins d'un certain niveau $j > i$ ont été rendus visibles à un niveau inférieur à i, et qu'aucun brin de niveau supérieur à j n'a été rendu visible à un niveau inférieur ou égal à i, alors la vue au niveau i est égale à la carte de niveau j ce qui signifie que la vue représente le même objet que la résolution j. Cette seconde propriété peut être formalisée ainsi :

Propriété 2 $\forall i \in [0,n]$, *si* $\exists\, j \geq i$ *tel que* $V^i = B^j$, *alors* $\forall k$, $\psi_k^i = \phi_k^j$.

Ces deux propriétés établissent une première forme de cohérence entre les vues et la carte multirésolution sous-jacente. Elles impliquent que lorsque l'ensemble des brins visibles à un niveau i correspond exactement à l'ensemble des brins d'un niveau de résolution j, alors les topologies de la vue et de la carte sous-jacente, observés à ces niveaux éventuellement différents, sont les mêmes. Cependant, l'intérêt du concept de vue réside dans sa capacité à représenter des objets intermédiaires composés de cellules appartenant à différents niveaux de résolution, tout en maintenant la cohérence de la topologie générée. D'autres contraintes sont ainsi nécessaires.

Un volume est dit *visible* au niveau i lorsque l'ensemble de ses brins le sont. Notons que ces brins peuvent avoir des niveaux d'insertion et/ou de visibilité très différents les uns des autres en fonction du niveau de subdivision et du niveau de résolution observé. Pour que les relations d'adjacence entre volumes soient cohérentes dans les vues, il faut s'assurer que la relation ψ_3 soit bien une involution entre brins visibles. Cela s'exprime avec la contrainte suivante : si un brin d est visible, alors $\phi_3^l(d)$ où l est le niveau d'insertion de d, doit également être visible :

Propriété 3 $\forall d, v(\phi_3^{niveau(d)}(d)) \leq v(d)$

Le même principe s'applique aux faces. Une face est visible si l'ensemble de ses brins le sont. Nous nous assurons que ψ_2 soient une involution entre brins visibles, avec la contrainte suivante :

Propriété 4 $\forall d, estVisible(d) \implies estVisible(\phi_2^{niveau(d)}(d))$

Il n'y a pas de contraintes aussi forte au niveau des arêtes car le côté adaptatif de la visibilité peut conduire à des situations ou certaines des faces incidentes à une arête peuvent être visibles alors que les autres ne le sont pas.

Dans la section suivante, nous expliquerons en détail comment les ensembles de brins visibles sont construits et comment les relations ψ_j^i sont mises à jour pour permettre l'ajout de nouveaux détails à une vue.

3.3 Gestion de la visibilité

Nous nous intéressons d'abord à la construction des ensembles de brins V^i, c'est-à-dire à la façon dont les brins seront sélectionnés dans la hiérarchie afin d'être rendus visibles. Notre objectif est d'ajouter des détails localement sur l'objet, en respectant les contraintes imposées plus haut.

Lorsque les brins d'une cellule sont rendus visibles au niveau i, nous parlerons de l'activation d'une cellule au niveau i. Cela englobe l'activation d'arêtes, de faces ou de volumes. Dans notre contexte de cartes multirésolutions, l'activation de cellules travaille sur les mêmes brins que les algorithmes de subdivision présentés auparavant. Nous illustrons cette notion sur deux exemples.

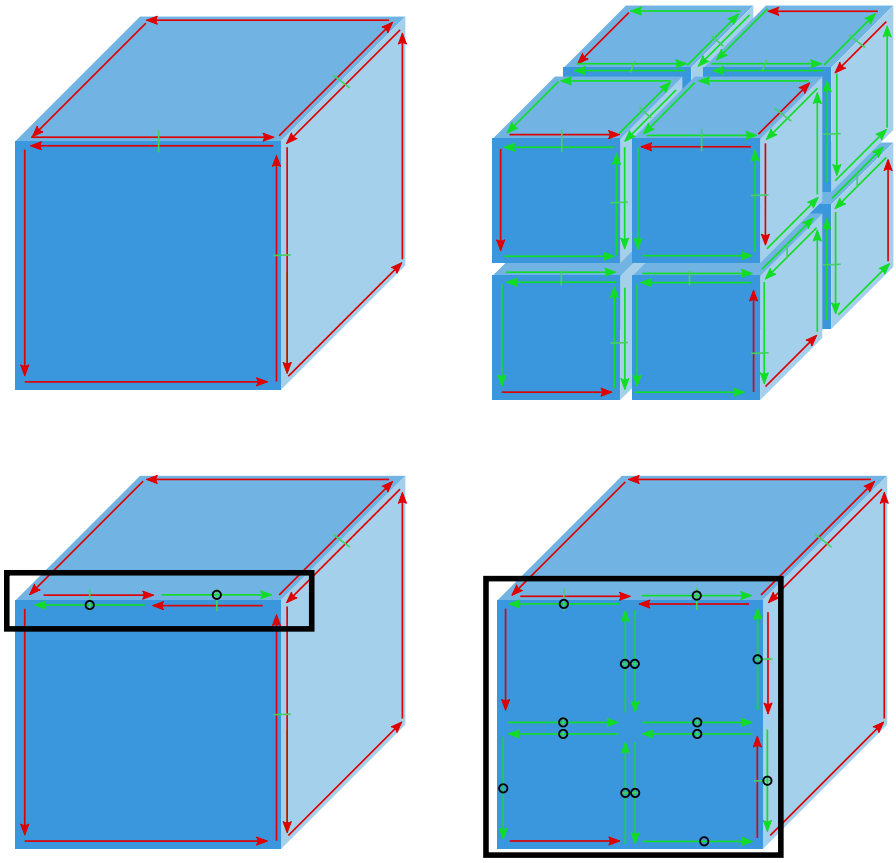

FIGURE 3.2 – En haut : deux niveaux de résolution d'un cube. Les flèches rouges et vertes représentent les brins de niveau 0 et de niveau 1. En bas : deux vues adaptatives où un sommet (à gauche) et 5 sommets (à droite) sont rendus visibles. Les brins de niveau 1 visibles au niveau 0 sont mis en évidence par des zéros. On dira qu'une arête (à droite) ou une face (à gauche) ont été activées au niveau 0.

Exemple 1 : activation d'une arête

Considérons la carte multirésolution de la Figure 3.2, contenant deux niveaux de résolution tels que le niveau 0 contienne 8 sommets formant un cube et le niveau 1 contienne 27 sommets formant 8 cubes résultants de la subdivision du cube de niveau 0. Activer une arête au niveau 0 consiste à rendre visibles les brins (dessinés en vert) de ses deux sous-arêtes qui n'existent qu'au niveau 1. Les deux sous-arêtes sont mises en évidence en bas à gauche de la Figure 3.2. On obtient ainsi une vue au niveau 0 avec 9 sommets (8 sommets du niveau 0 et 1 sommet du niveau 1).

Exemple 2 : activation d'une face

Dans la même carte, l'activation d'une face au niveau 0 consiste à rendre visibles les brins de ses 4 sous-faces. Ces sous-faces sont mises en évidence en bas à droite de la Figure 3.2 où la vue contient 13 sommets (8 sommets du niveau 0 et 5 du niveau 1).

3.3.1 Niveau d'une cellule

Notre objectif est de définir un algorithme permettant d'activer une k-cellule, pour $k > 0$, c'est-à-dire de rendre visible les brins correspondant à sa subdivision dans la hiérarchie multirésolution. Pour déterminer l'ensemble des brins à rendre visibles, nous devons trouver le niveau actuel de la cellule, nous avons déjà défini le niveau d'une cellule dans une carte multirésolution la définition pour une vue est similaire :

Définition 12 *Soit $cell_k^j(\mathcal{M}, d)$ et $cell_k^i(\mathcal{V}, d)$ les brins des k-cellules associés à d dans la résolution j de la carte multirésolution $\mathcal{M}$ et dans la résolution i de la vue $\mathcal{V}$.*

Le niveau d'une k-cellule de la vue $\mathcal{V}$ associée à un brin d pour $k \geq 2$ est le plus petit niveau j tel que $cell_k^j(\mathcal{M}, d) \subset cell_k^i(\mathcal{V}, d)$.

On note $niveau_k(\mathcal{V}, d)$ ce niveau ou $niveau_k(d)$ dans les situations où la vue n'a pas besoin d'être précisée.

La différence est que contrairement aux cartes multirésolution, cette définition ne s'applique pas aux arêtes, car une arête sur une face subdivisée d'un volume non subdivisé peut ne pas être entièrement visible en raison de faces non activées à l'intérieur du volume, comme illustré dans la figure 3.3. Pour une arête, le niveau de l'arête est défini comme étant le maximum entre le niveau d'insertion de d et de $\psi_2(d)$.

3.3.2 Activation des cellules

Dans la suite, nous détaillons les opérations d'activation d'arêtes, de faces et de volumes, ainsi que les opérations de désactivation correspondantes. La Figure 3.4 présente ces trois opérations. Le premier objectif est de déterminer l'ensemble des brins à rendre visibles pour activer une k-cellule c associée à un brin d. Ce sont les brins marqués de zéro sur la Figure 3.4. Pour y parvenir, nous avons besoin de déterminer les brins de niveau $niveau_k(d) + 1$ présents dans l'orbite des sous-cellules de c.

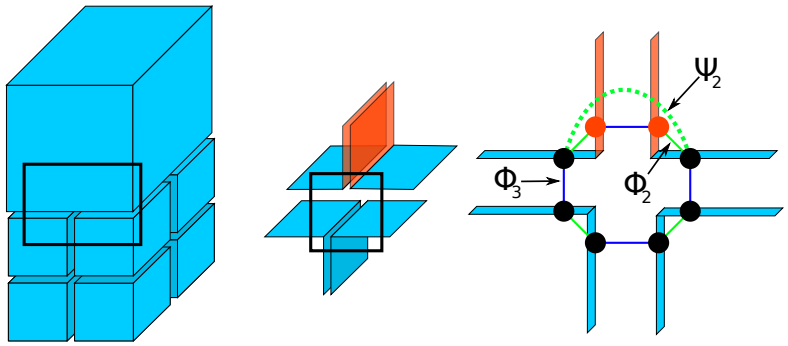

FIGURE 3.3 – Un volume de niveau 0 adjacent à quatre volumes de niveau 1 issus de la subdivision d'un volume de niveau 0. Nous nous intéressons aux arêtes sur la surface de la face de contacte entre ces volumes (fenêtre noire). Sur le zoom au milieu et à droite, on voit qu'une face interne du volume du haut (en rouge) n'est pas visible, l'orbite de l'arête doit donc le prendre en compte.

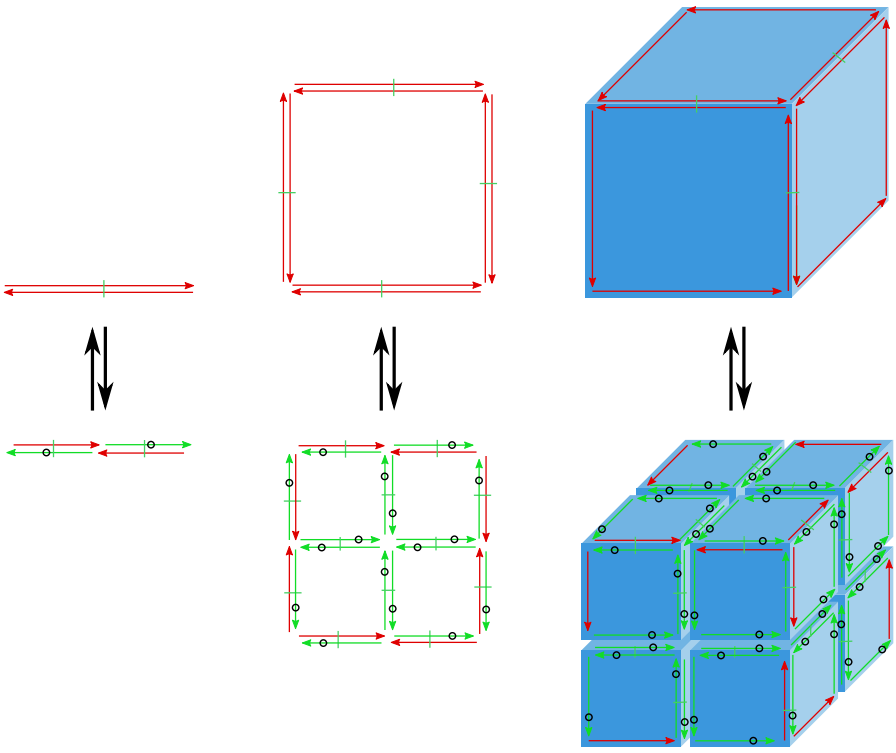

FIGURE 3.4 – Les trois opérations atomiques pour la gestion des vues : (de gauche à droite) activation et désactivation d'arête, de face et de volume.

Comme la hiérarchie multirésolution est construite par subdivision, nous savons que chaque sous-cellule de c partage un sommet avec c. Pour accéder aux sous-cellules, il suffit donc pour chaque sommet de c de sélectionner un brin d du sommet appartenant à c et de parcourir l'orbite de la k-cellule au niveau $niveau_k(d)+1$. Durant ce parcours, les sous-cellules sont activées en rendant visibles les brins de niveau $niveau_k(d)+1$ visités. Pour une cellule de dimension supérieure ou égale à 2, avant d'activer ses sous-cellules, il faut activer les cellules de son bord si elles ne le sont pas déjà. Il s'agit des arêtes d'une face ou des faces d'un volume. L'algorithme 2 décrit cette procédure d'activation.

Algorithme 2 *Activation*(k, d, i) : activation de la k-cellule de d au niveau i

```
l ← niveau_k(d)
c ← cell_k^l(d)
pour tout (k − 1)-cellules c' sur le bord de c faire
    si niveau_(k−1)(c') = l alors
        Activation(k − 1, c', i)
    fin si
fin pour
pour tout sommet s de c faire
    Choisir d' un brin de s appartenant à c
    pour tout brin d'' dans cell_k^(l+1)(d') faire
        si insersionLevel(d'') = l + 1 alors
            Visibility(d'') ← i
        fin si
    fin pour
fin pour
```

Cet algorithme fonctionne de manière efficace pour mettre à jour l'attribut de visibilité des brins concernés par une activation de cellule. Cependant, la Figure 3.5 illustre un cas limite où cet attribut de visibilité seul ne suffit pas à garantir une cohérence sur la connectivité des brins. Lorsqu'une face d'un volume est activée, une partie de l'orbite des arêtes à sa surface se trouve à l'intérieur du volume et ne doit pas être activée, comme le montre la Figure 3.3. Cela pose un problème lors de la découpe de l'une de ces arêtes : les brins en bleu sur la figure ne doivent pas être accessibles, alors que l'algorithme précédent les rend visibles. Il n'est pas possible de simplement ne pas mettre à jour leur visibilité, car lors d'une activation future du volume, ils devront faire parti de l'ensemble des brins de l'objet.

En utilisant la propriété 3, la visibilité d'un brin est désormais également déterminée par la visibilité du brin auquel il est lié à la résolution à laquelle il a été inséré, comme décrit dans l'algorithme 3. La vérification de cette propriété nous permet ainsi de résoudre ce problème sans modifier l'algorithme d'activation des brins.

Les algorithmes que nous venons de présenter présentent l'avantage d'afficher une complexité qui ne dépend pas du nombre de brins présents dans la carte. La complexité de l'algorithme d'activation de cellule est linéaire par rapport au nombre de brins présents

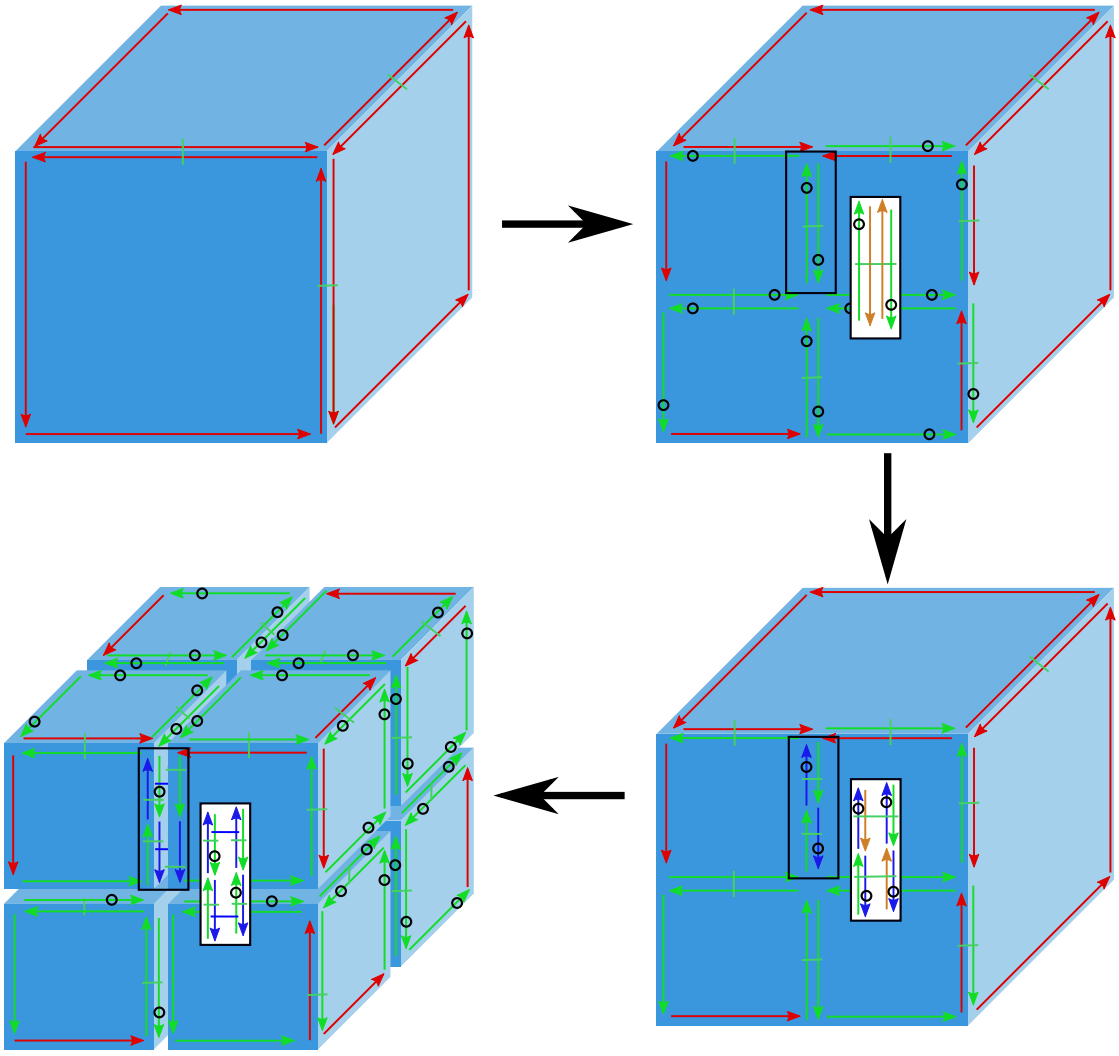

FIGURE 3.5 – Application successive d'une activation de face, une activation d'arête et une activation de volume, les cadre blanc permettent de voir les brins non activés des orbites d'arête représentée en orange.

Algorithme 3 $estVisible(d, i)$: détermine si d est visible au niveau i

1: **si** $Visibility(d) > i$ **alors** ▷ Si le brin n'est pas visible on renvoie Faux
2: **retourne** $False$
3: **fin si**
4: $l \leftarrow insersionLevel(d)$
5: $d3 \leftarrow \phi_3^l(d)$
6: **si** $insersionLevel(d3) < l$ **alors** ▷ Sinon on vérifie la visibilité de $\phi_3^l(d)$
7: **retourne** $estVisible(d3)$
8: **fin si**
9: **retourne** $True$ ▷ Ici $insersionLevel(d3) = l$

dans la cellule activée. La complexité de l'algorithme de détermination de la visibilité d'un brin est linéaire par rapport au nombre de niveaux de résolution présents.

L'activation de cellule permet de pouvoir ajouter des détails à un niveau de résolution donnée, en rendant visibles des cellules de manière isolée et adaptative. Il est par exemple possible d'activer les faces sur le bord d'un objet pour travailler sur les sommets utiles pour le rendu visuel ou d'activer des volumes fortement déformés dans la cadre d'une simulation physique. Nous allons maintenant aborder les algorithmes permettant de désactiver des cellules en modifiant la visibilité de leurs brins.

3.3.3 Désactivation des cellules

Nous appelons *désactivation d'une cellule* le processus qui rend invisibles les brins des sous-cellules de niveau l précédemment rendus visibles par l'activation d'une cellule de niveau $l-1$. Lors de la désactivation, les points suivants sont pris en compte :

1. Si une cellule du bord de la cellule désactivée est incidente à d'autres cellules elles-mêmes activées, elle ne doit pas être désactivée. Les cellules concernées pourront l'être ultérieurement lorsque les cellules incidentes seront désactivées.
2. Pour une cellule de dimension $k < n$ avec n la dimension de la carte combinatoire, il est nécessaire de vérifier que toutes les cellules incidentes de dimension supérieure à k ont un niveau inférieur ou égal à $l-1$. Par exemple avant de simplifier une face il faut vérifier que les deux volumes incidents soient bien à un niveau inférieur à celui de la face.
3. Si des sous-cellules de niveau l de la cellule désactivée sont elles-mêmes activées, il est nécessaire de les désactiver auparavant, de manière récursive.

L'algorithme 4 détaille la désactivation d'une cellule. Sa complexité est la même que pour l'activation des cellules.

Nous avons vu comment modifier la visibilité des brins pour activer ou désactiver des cellules au sein d'une vue et ainsi adapter localement les détails visibles à un niveau de résolution choisi. Nous allons maintenant voir comment reconstruire les relations entre ces brins de manière cohérente pour que les vues correspondent bien à des cartes combinatoires.

Algorithme 4 Désactivation(c) Désactivation d'une k-cellule

```
1: l ← niveau(c)
2: pour tout (k+1)-cellule adjacentes c_adj faire                ▷ Vérification de (1)
3:     si niveau(c_adj) ≠ l − 1 alors retourne
4:     fin si
5: fin pour
6: Soit c_1 la cellule au niveau l − 1
7: pour chaque sous cellule c_2 de c_1 faire
8:     si niveau(c_2) ≠ l alors                                  ▷ Vérification de (3)
9:         désactivation(c_2)
10:    fin si
11:    pour chaque brin b de c_2 dans la vue faire
12:        mark(b)
13:    fin pour
14: fin pour
15: S ← ∅
16: pour chaque sous cellule c_2 de c_1 faire
17:    pour chaque brin b de c_2 dans la vue faire
18:        si isMark(b) ∧ isMark(φ_k(b)) alors                   ▷ Vérification de (2)
19:            S ← S + b
20:        fin si
21:    fin pour
22: fin pour
23: pour chaque brin b dans S faire
24:    Visibilite(b) ← l
25: fin pour
```

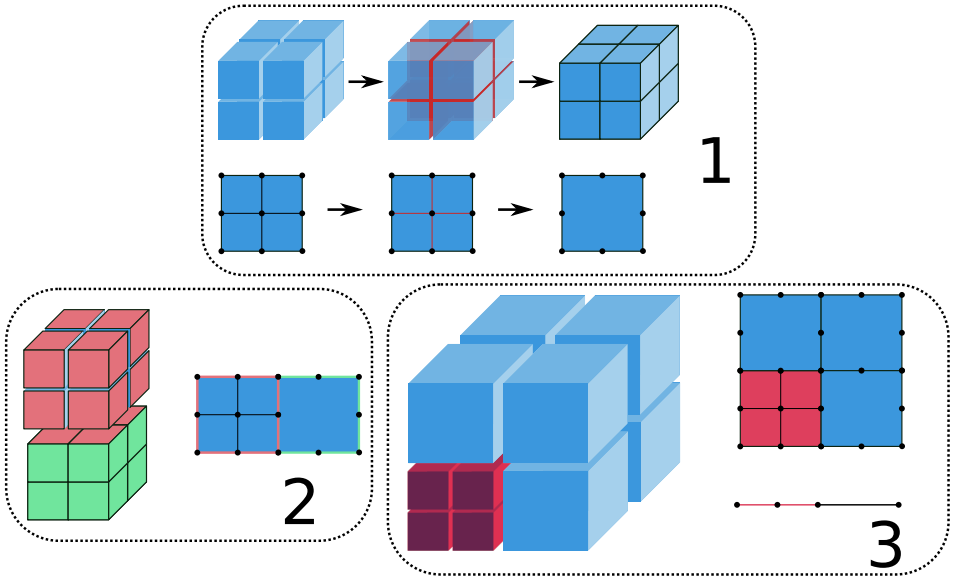

FIGURE 3.6 – Désactivation d'une cellule : (1) modifier la visibilité des brins des sous-cellules (en rouge) ; (2) désactiver les cellules du bord si les cellules adjacentes ne sont pas activées (action possible en vert, impossible en rouge) ; (3) désactiver les sous-cellules de niveau supérieur à l.

3.4 Reconstruction des relations d'incidence

Dans cette section, nous expliquons comment il est possible de reconstruire à la volée les relations ψ_i^k d'une vue. Pour cela nous proposons une définition des ψ_i^k à partir des relations ϕ_j^k stockées dans la carte multirésolution. Les algorithmes présentés ci-après exploitent la structure de 3-cartes multirésolutions construites par subdivision et sont donc spécifiques à la dimension 3 et au type de hiérarchies, construites par subdivision, que nous exploitons. Les vues doivent être des 3-cartes valides et donc ψ_3 et ψ_2 doivent être des involutions et ψ_1 une permutation avec la contrainte que $\psi_1 \circ \psi_3$ doit être une involution.

Nous commençons par définir ψ_3^i qui est la plus simple à traiter. La Figure 3.7 illustre le fait que $\psi_3^i(d)$ correspond à $\phi_3^i(d)$ la plupart du temps, sauf lorsque le brin d fait partie d'une arête activée. Dans ce cas, comme par construction, nous savons que ψ_3^i correspond à un ϕ_3^j mais que nous ne pouvons pas savoir quel est ce niveau j il faut monter dans les niveaux de résolution pour trouver le niveau le plus fin d'activation, c'est-à-dire le dernier niveau k où $\phi_3^k(d)$ est visible.

Il suffit pour cela de tester $\phi_3^j(d)$ pour $j > insertionLevel(d)$. Pour que nous puissions reconstruire son image à travers ψ_3^i, il est essentiel que le brin d soit visible au niveau i. L'algorithme 5 détaille la reconstruction de ψ_3^i, dans lequel nous utilisons directement l'attribut de visibilité du brin comme condition d'arrêt de la boucle.

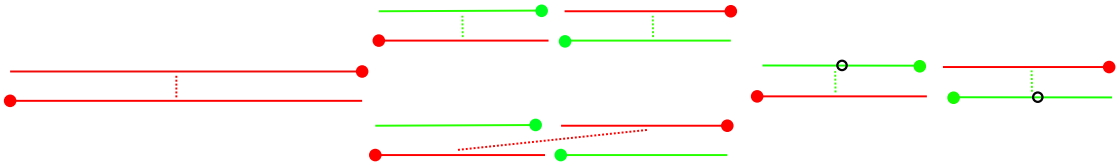

FIGURE 3.7 – Illustration de ψ_3^i. À gauche : une arête au niveau 0. Au centre : la même arête subdivisée au niveau 1. Les lignes pointillées représentent ϕ_3^0 (en bas) et ϕ_3^1 (en haut). À droite : les brins verts du niveau 1 sont rendu visibles au niveau 0. Les lignes pointillées vertes montrent les relations ψ_3^0 qui diffèrent de ϕ_3^0 et sont ici égales à ϕ_3^1

Algorithme 5 $\psi_3^i(d)$

$k \leftarrow niveauMaximum$
$it \leftarrow \phi_3^k(d)$
tant que $Visibilite(it) > i$ **faire**
  $k \leftarrow k - 1$
  $it \leftarrow \phi_3^k(d)$
fin tant que
retourne it

Cette approche repose sur l'hypothèse que ψ_3^i est appelé sur un brin déjà visible, ce qui évite l'exécution d'un parcours récursif coûteux de la fonction estVisible, en particulier dans une fonction aussi cruciale. Notons que la boucle s'arrête forcément par la propriété 3 qui garantit que $estVisible(\phi_3^{niveau(d)}(d))$ est vrai si d est visible au niveau i.

La complexité de la reconstruction de ψ_3^i est linéaire par rapport au nombre maximal de niveaux de résolution. Le parcours des niveaux se fait du niveau le plus fin au niveau le plus grossier, ce qui permet d'optimiser cette reconstruction pour les niveaux les plus fins qui sont aussi ceux qui contiennent le plus de brins.

Nous passons maintenant à ψ_2^i. L'image $\psi_2^i(d)$ d'un brin d peut être obtenue la plupart du temps à partir de $\phi_2^i(d)$ sauf dans certaines configurations où des faces ont été activées dans l'arête de d alors que d'autres sont restées non visibles. La Figure 3.8 illustre ce cas et montre que pour trouver $\psi_2^i(d)$ il faut *sauter* par-dessus les faces non visibles représentées en orange.

Pour trouver $\psi_2^i(d)$, il faut ainsi parcourir l'orbite d'arête au bon niveau dans la hiérarchie jusqu'à trouver un brin visible. Le niveau k à parcourir est le niveau d'insertion de la face de d, déterminé en cherchant le maximum des niveaux de d et de $\psi_3(d)$ que nous venons de définir. L'orbite arête est ensuite parcourue en commençant par $\phi_2^k(d)$ puis en sautant les faces invisibles en suivant la permutation $\phi_2^k \circ \phi_3^k$ jusqu'à trouver un brin visible. L'algorithme 6 détaille la reconstruction de ψ_2. Sa complexité est celle de ψ_3 pour déterminer le niveau de l'arête, et dépend du nombre de brins dans l'orbite d'arête. Comme pour ψ_3 cette complexité dépend uniquement du voisinage local et non de la taille de la carte.

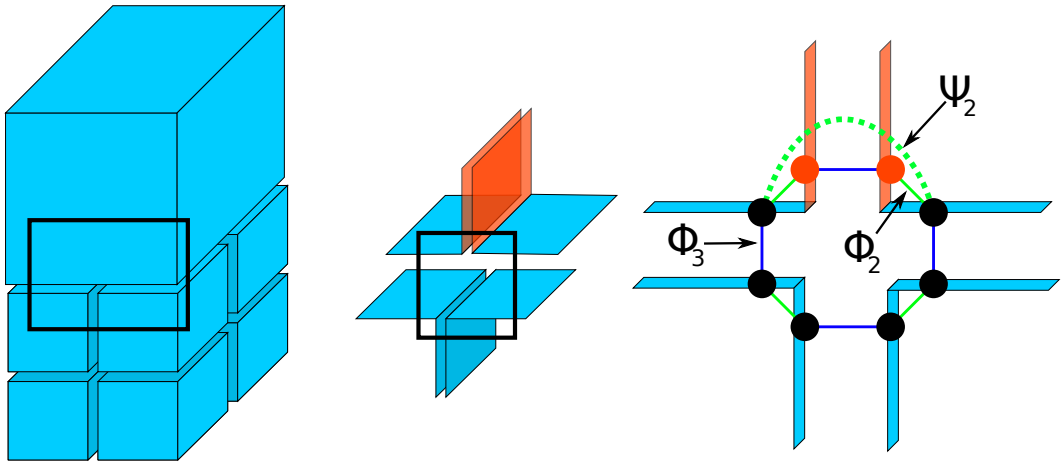

FIGURE 3.8 – Illustration de ψ_2. La face rouge n'est pas visible au niveau 0 mais existe au niveau 1, alors que les faces adjacentes sont visibles au niveau 0, leur face parente ayant été activée. L'image par ψ_2 est obtenue en cherchant le premier brin visible dans l'orbite d'arête. La courbe verte montre le chemin emprunté.

Algorithme 6 $\psi_2^i(d)$

$e \leftarrow \psi_3(d)$
$k \leftarrow max(niveau(d), niveau(e))$
$it \leftarrow \phi_2^k(d)$
tant que $!estVisible(it)$ **faire**
  $it \leftarrow \phi_2^k(\phi_3^k(it))$
fin tant que
retourne it

Nous terminons avec la relation ψ_1 dont la reconstruction est la plus complexe. Comme pour les autres relations, ψ_1 correspond à une des relations ϕ_1^j uniquement si la face ou l'arête du brin n'ont pas été activées. Dans le cas contraire, il faut parcourir le niveau de résolution où l'activation la plus fine a été effectuée. À ce niveau, certaines arêtes ont pu être activées sans que les faces incidentes le soit, dans ce cas pour trouver l'image d'un brin par ψ_1 il faut *sauter* les cellules du niveau parcouru qui ne sont pas visibles.

La Figure 3.9 montre comment les relations ϕ_1^i s'imbriquent sur trois niveaux de résolution. Ce sont typiquement les arêtes intermédiaires qui doivent être sautées pour reconstruire les ψ_1^i. Notons également que les faces parcourues peuvent l'être de manière horaire ou anti-horaire avec ϕ^i_{-1}. Enfin, rappelons que la contrainte stipulant que $\phi_1^i \circ \phi_3^i$ est une involution implique que les relations ϕ_1^i de faces de volumes adjacents sont symétriques les unes des autres. L'algorithme de reconstruction utilise cette propriété en passant d'un volume à l'autre pour optimiser le nombre de sauts à effectuer.

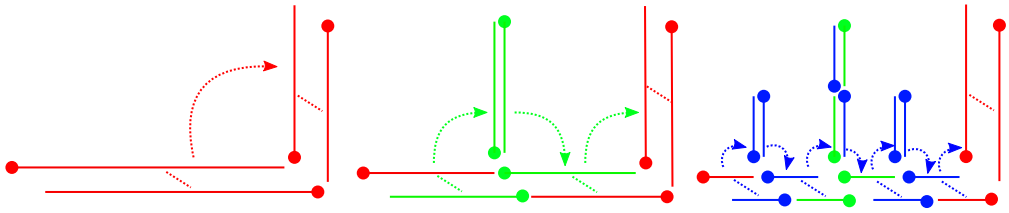

FIGURE 3.9 – Illustration de la relation ϕ_1 pour 3 niveaux de résolution : ϕ_1^0 (en rouge), ϕ_1^1 (en vert) et ϕ_1^2 (en bleu). Deux faces de volumes adjacents sont (partiellement) dessinées pour mettre en évidence le fait qu'elles peuvent être traversées dans les sens horaire et anti-horaire en fonction du volume considéré. La relation ϕ_3^i permet de passer d'un volume à l'autre.

Il existe trois cas différents pour déterminer $\psi_1^i(d)$. Ces cas sont illustrés dans la Figure 3.10 et détaillés dans l'algorithme 7. Pour distinguer ces différents cas, nous exploitons le fait que, par construction, dans un schéma de subdivision primale, tous les brins d'une orbite de sommet ont le même niveau d'insertion. Cela signifie que pour un brin d, le niveau de $\psi_1(d)$ est égal au niveau de $\psi_3(d)$ et $\psi_2(d)$. Nous comparons donc les niveaux réels de d et de $\psi_3^i(d)$ pour définir ces cas.

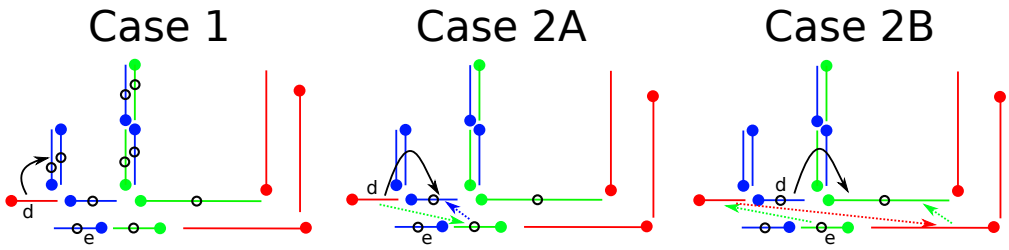

FIGURE 3.10 – Illustration de la reconstruction de ψ_1^i et des trois situations traitées dans l'algorithme 7. Les brins rendus visibles au niveau 0 sont marqués d'un 0. À gauche : on part du brin d au niveau 0. Le brin $e = \psi_3^0(d)$ est de niveau 2, et $\phi_1^2(d)$ est visible au niveau 2. Au centre : les mêmes brins d et e mais $\phi_1^2(d)$ n'est pas visible. Le *saut* en noir se fait en utilisant l'arête de niveau 1, ce qui est illustré par les flèches en pointillés verts et bleus. À droite : cette fois-ci, le niveau du brin d est 2 et celui du brin e est 1. Le saut se fait en 4 étapes.

Le premier cas correspond à la situation où $\phi_1^l(d)$ est visible, avec l étant le maximum entre le niveau de d et le niveau de $\psi_3(d)$. C'est le cas où ψ_1 correspond à un ϕ_1^j de la hiérarchie multirésolution. Cette situation se produit lorsque le niveau de l'arête de d correspond au niveau de la face de d. Les deux cas suivants correspondent à la situation où $\phi_1^l(d)$ n'est pas visible. Pour trouver $\psi_1(d)$, il faut effectuer un *saut* soit en passant au-dessus des arêtes non visibles du sommet, soit en parcourant les brins de l'arête à différentes résolutions.

Algorithme 7 $\psi_1^i(d)$

$e \leftarrow \psi_3^i(d)$
$l \leftarrow niveau(e)$
$l_d \leftarrow niveau(d)$
/* Case 1 */
si $estVisible(\phi_1^{max(l,l_d)}(d))$ **alors retourne** $\phi_1^{max(l,l_d)}(d)$
fin si
/* Case 2 */
si $l_d < l$ **alors**
    /* Case 2A */ **retourne** $\phi_3^l(\phi_3^{l-1}(d))$
fin si
si $l_d > l$ **alors**
    /* Case 2B */ **retourne** $\phi_3^l(\phi_3^{l-1}(\phi_3^l(e)))$
fin si

Pour distinguer entre ces deux cas, on examine le niveau de $\psi_3(d)$. Si ce niveau est supérieur à celui de d, alors $\psi_1(d) = \phi_3^l(\phi_3^{l-1}(d))$. On utilise la hiérarchie multirésolution, pour sauter le sommet de niveau l en une seule étape quelque soit le degré de ce sommet. Si le niveau de $\psi_3(d)$ est inférieur à celui de d, alors $\psi_1(d) = \phi_3^l(\phi_3^{l-1}(\phi_3^l(\psi_3^i(d))))$. Enfin, si le niveau de $\psi_3(d)$ est égal à celui de d, nous sommes sur une arête non activée, et nous revenons donc au cas trivial. Ces cas correspondent à la situation où d se trouve sur une arête subdivisée d'une face non subdivisée. Toutes ces situations sont illustrées dans la Figure 3.10. L'évaluation de ces différents cas se fait en temps constant. La complexité de la reconstruction de ψ_1 est donc celle de ψ_3, qui est liée, comme nous l'avons vu, au nombre total de niveaux de résolution.

Grâce aux opérations d'activation de cellules et aux algorithmes de reconstruction associés que nous avons présentés, nous avons pu définir des vues multirésolution adaptatives. Ces vues permettent de générer à la volée une hiérarchie des 3-cartes à partir d'une hiérarchie construite par subdivision et d'ensembles de brins rendus visibles par l'activation de cellules à n'importe quel niveau de résolution. Dans la suite nous allons affiner ce modèle en y apportant quelques compléments et en étudiant son implantation et ses performances dans le cadre d'application de simulation interactives.

3.5 Hiérarchie de vues

Une unique hiérarchie de 3-cartes multirésolutions peut être utilisée pour représenter un grand nombre de vues. Dans ce cas, chaque vue doit simplement stocker l'information de visibilité des brins. Les relations topologiques sont reconstruites à la volée et ne nécessitent aucun stockage. Les différentes vues peuvent être exploitées avec des objectifs distincts dépendant des choix de développement comme rendre visible des sommets de la surface pour permettre un rendu adaptatif ou rendre visibles des sommets internes pour mieux représenter une découpe ou une déchirure.

Une amélioration simple et particulièrement utile consiste à combiner plusieurs vues pour former des hiérarchies de vues. Pour cela, nous définissons une relation d'héritage entre deux vues $\mathcal{V}_a^i$ et $\mathcal{V}_b^j$. Nous disons que $\mathcal{V}_b^i$ hérite de $\mathcal{V}_a^j$ si un brin visible dans $\mathcal{V}_a^i$ est également visible dans $\mathcal{V}_b^j$ quelque soit les informations de visibilités stockées dans les deux vues.

Pour mettre en place cette hiérarchie, nous proposons deux mécanismes. Le premier consiste à faire en sorte que les deux vues partagent le même attribut de visibilité, mais parcourent la hiérarchie à des niveaux différents. Cela signifie, comme indiqué dans l'algorithme 3, que tout brin visible dans la vue de niveau inférieur est automatiquement visible dans la vue de niveau supérieur.

Le second mécanisme consiste à utiliser deux vues possédant des attributs de visibilité distincts, sans contrainte sur leur niveau. Lors de la vérification de la visibilité d'un brin dans la vue fille, il suffit de vérifier si ce brin est également visible dans la vue mère. Grâce à ces mécanismes, des hiérarchies de vues peuvent être construites en combinant des critères d'adaptation et des niveaux de résolution de manière précise et subtile. La synchronisation de la visibilité entre les vues est immédiate et ne nécessite pas de calcul supplémentaire.

3.6 Découpe et vue topologique

Une opération importante tirant parti de ce mécanisme de hiérarchie de vues est la découpe d'un objet multirésolution. Si on s'intéresse à l'aspect topologique d'une découpe sans se préoccuper des problèmes de plongement physique ou géométrique, alors réaliser une découpe revient simplement à séparer certains volumes au sein d'un objet.

Dans une carte combinatoire, les découpes sont gérées en modifiant le lien ϕ_3 entre les brins. Pour respecter les contraintes d'intégrité, deux volumes sont toujours séparés le long de faces entières. Ainsi, l'opération consiste à séparer deux faces en modifiant simultanément le lien ϕ_3 de leurs brins. Lorsqu'un ensemble de faces, deux à deux adjacentes et formant un cycle fermé autour d'un sommet, est séparé, le sommet est naturellement divisé en deux sommets déconnectés sans traitement supplémentaire. Ce mécanisme de découpe peut être précédé d'opérations de remaillage local et/ou suivi d'un ajustement ou d'autres techniques de lissage, que nous aborderons dans un futur chapitre.

Dans une carte multirésolution, la découpe est effectuée de façon similaire et indépendante sur chaque niveau de résolution. Pour assurer une certaine cohérence entre les niveaux de résolution, nous imposons le fait que si deux faces sont séparées, alors toutes leurs sous-faces dans la hiérarchie doivent également être séparées.

Deux types de découpe peuvent être utilisés sur notre modèle : la découpe partielle d'une face, c'est-à-dire ne pas séparer l'ensemble des sous-faces d'une face pour représenter une découpe partielle, et la séparation de sous-volumes à l'intérieur d'un volume grossier. Ces opérations de découpe partielle ne posent pas de problème d'un point de vue topologique. Cependant, lorsqu'un sous-volume est séparé, il est incorrect d'un point de vue sémantique que ses volumes parents soient toujours connectés, ou que lorsque deux sous-volumes sont déconnectés, leur volume parent soit encore parcouru.

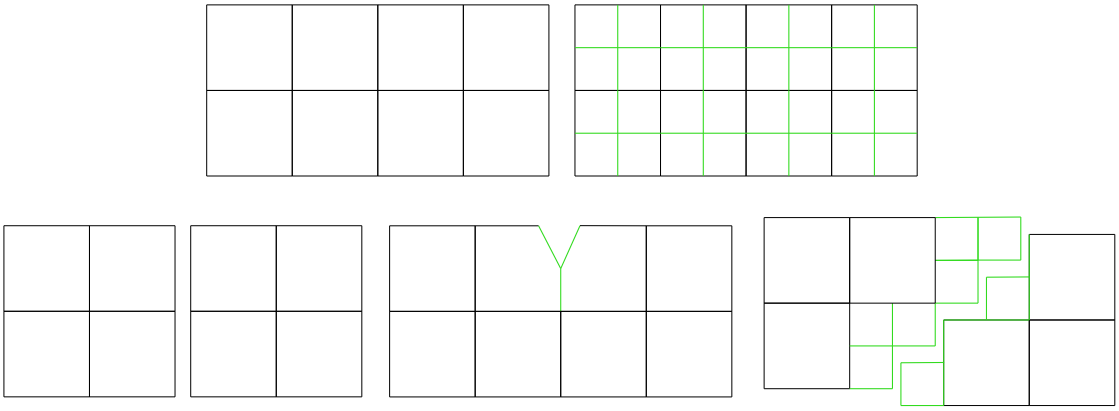

FIGURE 3.11 – Exemple des différentes situations rencontrées lors d'une découpe. En haut : deux niveaux de résolution d'un objet. En bas, de gauche à droite : séparation complète de deux cellules grossières, séparation partielle, séparation de sous-cellules. Les maillages représentés correspondent à la vue topologique associé à ces découpes.

La solution consiste à ce que les volumes fins le long de la découpe soient toujours visibles dans toutes les vues utilisées. Plutôt que d'utiliser un processus long et complexe d'activation explicite de volume dans toutes les vues de l'objet, nous proposons simplement de créer une "vue topologique" de niveau 0 placée tout en haut de la hiérarchie de vue, c'est-à-dire que toutes les vues héritent au minimum de cette vue. Les découpes seront ainsi réalisées sur cette vue, ce qui nous permet de résoudre le problème de volume devant être toujours visible.

En utilisant ce mécanisme de hiérarchie de vues, nous pouvons ainsi réaliser des découpes partielles complètes tout en garantissant une intégrité du bord de l'objet dans l'intégralité des vues de l'objet, et cela automatiquement et en un temps constant ne dépendant pas du nombre total de vues de l'objet.

3.7 Implantation

Nous allons maintenant détailler notre implantation du modèle des vues multirésolution et évaluer les performances de cette implantation. L'ensemble des développements ont été réalisés avec la librairie CGoGN qui propose une implantation particulièrement optimisée des cartes combinatoires monorésolutions.

Les vues multirésolution sont compatibles avec les représentations explicites ou implicites des cartes combinatoires. C'est pourquoi les deux versions ont été mises en œuvre dans un premier temps. L'utilisation des cartes multirésolutions explicites, plus performantes notamment parce qu'elles stockent directement les relations entre brins aux différents niveaux de résolution, a été privilégiée lors des différentes expérimentations.

La structure de données pour représenter une vue est très simple. Elle consiste en une référence sur une carte multirésolution, un pointeur vers une autre vue pour représenter

la hiérarchie de vues, pointeur pouvant être NULL si la vue n'hérite d'aucune autre vue, et enfin un attribut de brin permettant d'associer à chaque brin un entier correspondant à son niveau de visibilité. La création d'une nouvelle vue consomme donc un espace mémoire proportionnel au nombre de brins dans la carte multirésolution multiplié par la taille d'un entier en mémoire.

Dans un contexte où la place mémoire n'est pas critique et où les performances sont prioritaires, nous avons mis en place un système de cache permettant de stocker les relations ψ_i au niveau des brins, évitant de devoir recalculer plusieurs fois la même relation lorsqu'elle n'est pas modifiée. Un système d'horloge est associé à ce cache afin de savoir quand mettre à jour suite à une modification de la topologie de la carte lors d'une activation, de la désactivation de cellules ou la séparation de cellule.

3.7.1 Plongement de cellule

Les cartes combinatoires permettent d'associer à chaque type de cellule un attribut distinct. Dans CGoGN, cette gestion des attributs est réalisée en associant à chaque cellule un indice dans un tableau qui stocke ces attributs. Dans une carte combinatoire classique, cet indice est porté par les brins, chaque brin possédant ainsi $n+1$ indices de cellules pour une n-carte.

Pour les cartes multirésolutions, deux approches sont possibles. Dans une représentation explicite, chaque brin dispose de $n+1$ indices par niveau de résolution stockés explicitement. Dans une représentation implicite, chaque brin ne stocke que les $n+1$ indices de la résolution correspondant à son niveau d'insertion. Ainsi, pour une cellule donnée, il faut d'abord retrouver les brins les plus jeunes dans son orbite pour connaître son indice. Dans une vue multirésolution, le principe est le même : l'indice d'une k-cellule est porté par les brins du même niveau que le niveau de la cellule. Ceci permet de garantir que chaque cellule possède un indice unique.

L'utilisation d'une représentation implicite des plongements de cellules permet d'économiser de l'espace mémoire par rapport à une représentation explicite, car seuls les indices des cellules effectivement présentes dans la carte sont stockés. Cependant, cela peut entraîner une légère perte de performance due à la nécessité de retrouver les brins les plus jeunes dans l'orbite d'une cellule pour obtenir son indice. De la même façon que pour les relations entre brin un système de cache d'indice de cellule peut être mis en place si les performances sont critiques pour l'application.

3.8 Performance des vues multirésolution

Les vues adaptatives permettent un accès efficace et concurrent aux sommets de différents niveaux de résolution au sein des différentes boucles de traitements. Pour démontrer la pertinence de cette approche, nous allons d'abord comparer les performances des vues à celles des maillages classiques.

Pour ce faire, nous avons mis en place une série d'expériences comprenant la traversée d'éléments et de sommets actifs, l'accès aux cellules adjacentes et des calculs similaires

à ceux utilisés dans les logiciels d'animation habituels. Nous avons choisi des opérations géométriques simples de sorte que les temps de calcul observés reflètent correctement les performances des structures de données et ne soient pas altérés par la mise en œuvre des calculs numériques impliqués. Quatre opérations sont implémentées : le calcul du centre de gravité d'un élément en parcourant ses sommets ; une opération de lissage consistant à placer un sommet au centre des sommets des volumes adjacents ; la subdivision d'un volume en 8 sous-volumes et la simplification d'un ensemble de volumes précédemment subdivisés. Ces deux dernières opérations sont celles qui sont nécessaires pour adapter les maillages de manière interactive.

Les structures de données classiques destinées à la gestion de maillages ne permettent pas d'annuler efficacement des opérations de subdivision / simplification répétées au fil du temps qui sont pourtant nécessaires à la gestion adaptative des sommets. Pour une comparaison équitable, il convient d'enrichir les représentations de maillage habituelles pour permettre une mise en œuvre efficace des opérations de simplification. Chaque fois qu'un volume est subdivisé en plusieurs sous-volumes, les relations hiérarchiques entre les parents et leurs enfants doivent être stockées pour permettre d'annuler efficacement cette opération de subdivision. En pratique, pour effectuer les comparaisons, nous utilisons le modèle CPH qui implémente une telle hiérarchie [Unt+15] car ils ont prouvé qu'ils avaient des performances similaires à celles des cartes de résolution unique lorsqu'il y a peu de niveaux de résolution.

3.8.1 Traversées d'une résolution unique

Tout d'abord, nous avons cherché à estimer le surcoût de notre structure multirésolution par rapport à une structure classique pour les opérations impliquant une seule résolution. Nous avons effectué les opérations de centroid et de lissage sur l'ensemble du maillage et pour différents niveaux de résolution. Cette opération a été répétée des centaines de fois pour obtenir les temps de calcul moyens affichés dans la figure 3.12.

Ces résultats sont conformes à ce qui était attendu, itérer à travers les volumes et les sommets dans des vues nécessite de tester leur visibilité, alors que pour les structures classiques, les accès sont directs. Cependant, nous constatons que les différences de performances diminuent pour les niveaux de résolution les plus fins qui contiennent le plus de sommets. Cela tempère les différences observées ici car les calculs sont coûteux pour les sommets de niveau 0 qui sont les moins nombreux. Les temps de calcul totaux étant fortement liés au temps requis pour les niveaux les plus fins qui contiennent, de loin, le plus de sommets.

3.8.2 Traversées adaptatives

Dans un second temps, nous avons basculé vers un contexte adaptatif. Nous avons commencé avec un maillage donné et effectué des cycles de subdivision, de calcul de centroides, de lissage et de simplification, tout en maintenant le nombre total de volumes sous une limite fixe. Le calcul est effectué, d'une part, avec une carte combinatoire et,

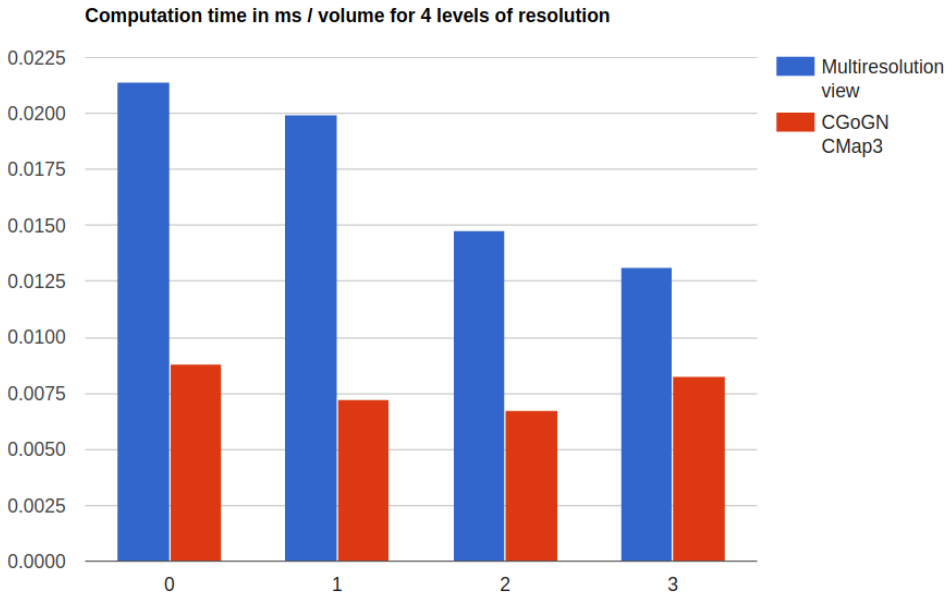

FIGURE 3.12 – Performances des vues multirésolution (en bleu) par rapport aux cartes combinatoires (en rouge), pour quatre niveaux de résolution avec respectivement 3,856, 30,848, 246,784 et 1,974,272 volumes.

d'autre part, avec une vue multirésolution, dont le niveau le plus fin contient le même nombre de volumes que la carte.

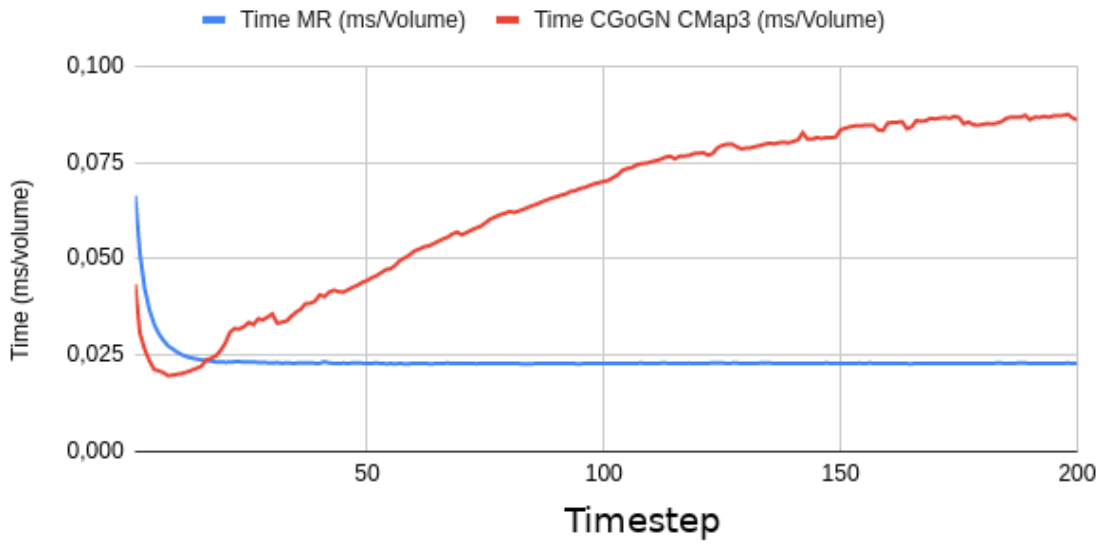

FIGURE 3.13 – Évolution des performances des cycles d'animation (subdivision, centroïde, lissage, simplification) pour une vue multirésolution (en bleu) et une carte combinatoire (en rouge). Le maillage initial contient 246 784 volumes et environ 1 million après quelques itérations.

Les performances obtenues sont présentées dans la figure 3.13. Nous pouvons voir clairement que les performances du modèle de résolution unique se dégradent après quelques cycles pour atteindre un plateau où les temps de calcul sont 4 fois supérieurs à ceux de nos vues multirésolution. La dégradation est due à la fragmentation des données en mémoire. Avec une structure de données classique, les cellules créées et supprimées au cours des phases d'adaptation sont progressivement placées de manière non contiguë en mémoire. Au contraire, avec les vues multirésolution, les données restent en place et seule l'information de visibilité est modifiée.

Certaines bibliothèques de gestion de maillage offrent des opérations de regroupement ou de réalignement des données. Nous avons implémenté un tel algorithme de réalignement pour les structures monorésolution de CGoGN et mené des tests de performance. La figure 3.14 montre l'évolution des temps de calcul avec un réalignement tous les 15 cycles. Cela améliore les performances globales du modèle monorésolution, mais entraîne de fortes oscillations des temps de calcul, ce qui est particulièrement critique pour les animations en temps réel. Les vues multirésolutions restent plus efficaces à long terme.

Les résultats présentés ci-dessus dépendent intrinsèquement du nombre de volumes qui sont subdivisés ou simplifiés à chaque étape. Pour illustrer ces différents scénarios, nous avons mis en place un dernier test dans lequel nous faisons varier le pourcentage

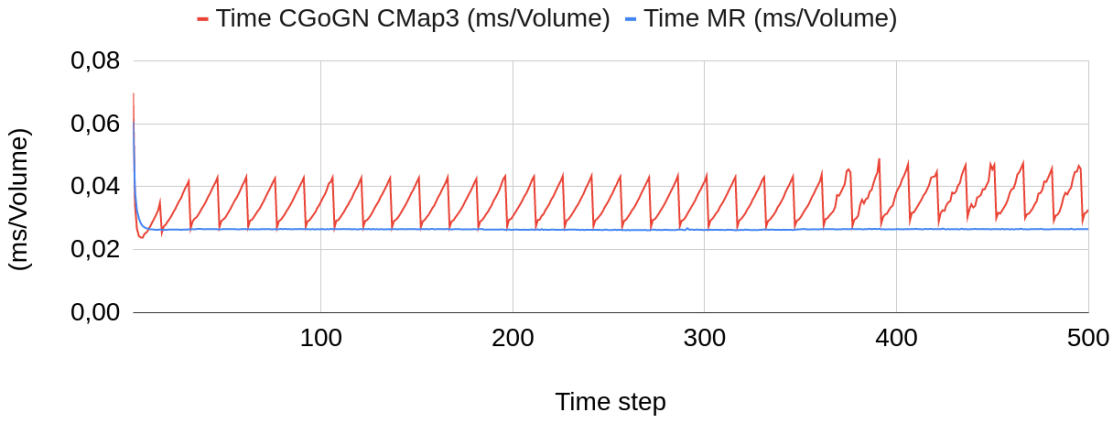

FIGURE 3.14 – Évolution des performances des cycles d'animation avec un compactage de données toutes les 15 itérations, pour des cartes combinatoires (en rouge). Les vues multirésolutions (en bleu) n'ont pas besoin d'un tel compactage.

de volumes subdivisés à chaque étape. Nous avons mesuré les performances des cycles d'animation en imposant la subdivision de 10, 20 et 30% des volumes et en simplifiant autant de volumes que nécessaire pour rester en dessous du seuil de 1 million de volumes.

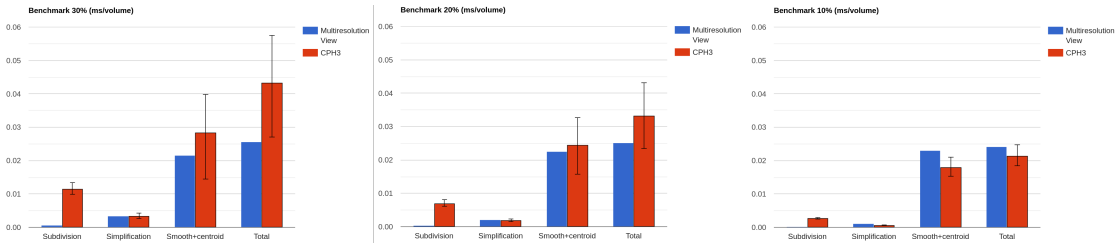

FIGURE 3.15 – Temps de calcul moyen pour trois scénarios où 10, 20 et 30% des volumes sont subdivisés à chaque pas de temps. Les temps pour la subdivision, la simplification, les opérations géométriques et leurs totaux sont présentés en bleu pour les vues multirésolution et en rouge pour les cartes. Les temps de calcul sont représentés avec une barre indiquant les valeurs minimale et maximale entre lesquelles ils oscillent.

La figure 3.15 montre le résultat de ces expériences. Elle détaille les temps de calcul moyens pour chaque étape du cycle d'animation, ainsi que les valeurs minimales et maximales entre lesquelles ces temps oscillent. Nous observons que le modèle de vue est, comme prévu, très efficace pour les opérations de subdivision et de simplification. Les temps de calcul pour les opérations géométriques montrent que plus le pourcentage de volume subdivisé est élevé, plus les oscillations sont importantes pour les modèles monorésolution.

À ce stade, il convient de noter que ces résultats ont été obtenus en n'utilisant que deux niveaux de résolution, c'est-à-dire qu'après une première subdivision, les sous-volumes résultants ne sont jamais subdivisés. Cette contrainte a été imposée par le modèle monorésolution utilisé et la hiérarchie adhoc mise en place pour permettre l'annulation de la subdivision et une comparaison équitable des deux modèles. Les vues multirésolutions supportent naturellement un nombre arbitraire de niveaux de résolution. L'augmentation du nombre de niveaux de résolution utilisés améliore l'adaptabilité de la distribution des sommets dans la simulation tout en bénéficiant de meilleures performances, ce qui est l'objectif de notre modèle.

En conclusion, les calculs *en place* qui conservent les alignements en mémoire et la gestion optimale des modifications topologiques font que les performances et la stabilité de temps de calcul observés sont largement supérieurs pour les vues multirésolution que pour une structure classique ce qui démontre l'intérêt de ce modèle.

3.9 Conclusion

Dans ce chapitre, nous avons présenté le modèle des vues multirésolution adaptatives. Ce modèle permet de gérer dynamiquement la répartition des détails au sein un objet en s'appuyant sur le modèle des cartes combinatoires multirésolution enrichi avec une information de visibilité.

Nous avons décrit ce modèle d'un point de vue formel, puis nous avons détaillé différents algorithmes permettant de l'exploiter. Nous avons ensuite expliqué comment créer des hiérarchies de vues liées par une relation d'héritage. Cela nous a permis de proposer une vue topologique, qui permet de gérer aisément les découpes sur notre hiérarchie de vues, ainsi que la cohérence du bord de notre objet.

Nous avons mis en place ce modèle dans le cadre d'une hiérarchie multirésolution obtenue par raffinement primal. Cela nous a permis d'exploiter un certain nombre d'hypothèses pour reconstruire les relations entre les brins et ainsi de créer un grand nombre de vues indépendantes et toutes synchronisées avec la hiérarchie sous-jacente, à moindre coût.

Enfin, nous avons démontré la pertinence de notre modèle en termes de performances par rapport à un modèle classique. Les données des différentes cellules ne bougent jamais en mémoire, ce qui permet d'optimiser les accès et d'amortir les coûts de reconstruction des relations entre les brins.

Deuxième partie

Animation interactive d'objets détaillés

Chapitre 4

État de l'art des méthodes d'animation

4.1 Animation d'objets déformables

4.1.1 Introduction

L'animation et la simulation d'objets déformables sont des thèmes de recherche importants en informatique graphique depuis de nombreuses années. Depuis les travaux fondateurs de Terzopoulos et al. en 1987 [Ter+87], de nombreuses recherches ont été menées pour améliorer le réalisme physique des animations créées en informatique. Les solides déformables sont des éléments centraux dans la création de modèles de simulation physiquement précis, et leur utilisation est devenue courante dans un grand nombre d'applications.

Les modèles de solides déformables se basent sur des concepts issus de la mécanique et des mathématiques appliquées, qui ont été adaptés pour une utilisation en informatique. Ces modèles permettent de simuler la déformation de solides en réponse à des forces externes, ce qui est crucial pour de nombreuses applications telles que la réalité virtuelle, la simulation de collisions ou encore la modélisation de tissus biologiques. En reproduisant une grande variété de comportements physiques tels que la torsion, la flexion et la compression, les modèles de solides déformables permettent de créer des animations plus réalistes.

Les avancées dans ce domaine peuvent être réparties selon deux axes : l'amélioration du réalisme physique des animations en complexifiant les modèles utilisés, d'une part, et l'optimisation algorithmique de ces modèles afin de les utiliser dans des applications temps réel ou en temps interactif, d'autre part.

Dans ce chapitre, nous allons commencer par rappeler les principes mécaniques de base qui sous-tendent la simulation d'objets déformables. Nous présenterons ensuite les principes des deux grandes familles de modèles : les modèles de mécanique discrète et les modèles de mécanique continue. Enfin, nous ferons un état de l'art rapide des modèles mécaniques disponibles dans la littérature.

4.1.2 Principe fondamentale de la mécanique

Approche lagrangienne et eulérienne

Pour étudier la déformation d'un corps, deux approches différentes existent dans la littérature : l'approche Lagrangienne et l'approche Eulérienne.

Approche lagrangienne : l'objet est discrétisé en un ensemble de particules mobiles, chacune étant caractérisée par une position de départ et dont la trajectoire est déterminée au cours du temps.

Approche eulérienne : les grandeurs physiques (scalaires, vecteurs, tenseurs) et leurs variations sont décrites en chaque point de l'espace. Pour cela, un certain nombre de points fixes sont choisis dans l'espace de l'objet et les propriétés mécaniques et leurs variations sont calculées uniquement en ces points. Cette approche est généralement utilisée pour les fluides.

Il convient de noter que l'approche eulérienne n'est pas couramment utilisée pour les solides déformables, où l'approche lagrangienne est largement préférée. Pour cette raison, dans la suite de ce manuscrit, nous ne nous intéressons qu'à la représentation des objets avec l'approche lagrangienne.

Dans l'approche lagrangienne, pour étudier la déformation d'un objet, nous avons besoin de le représenter par un ensemble de particules discrètes. Pour cela, nous pouvons utiliser un maillage comme décrit précédemment, où les sommets représentent les particules et les autres types de cellules et leurs relations d'incidence peuvent être utilisées par différents modèles mécaniques.

Principe fondamentale de la dynamique

L'objectif de l'animation d'objets solides déformables est de déterminer la position de chaque particule i, pour cela, nous nous appuyons sur la seconde loi de Newton, également connue sous le nom de principe fondamental de la dynamique (PFD) :

$$m_i \ddot{x}_i = F(x_i, \dot{x}_i, t) \tag{4.1}$$

Cette équation établit un lien entre l'accélération $\ddot{x}_i$ d'une particule, sa masse m_i, sa vitesse $\dot{x}_i$ et sa position x_i à chaque instant t. Pour trouver la position x_i, il est donc nécessaire de résoudre cette équation différentielle. Nous verrons par la suite différentes méthodes numériques permettant de la résoudre. L'objectif d'un modèle mécanique est de déterminer les forces appliquées sur chaque particule en réponse aux forces extérieures qu'elles subissent.

4.1.3 Mécanique des milieux continues

La mécanique des milieux continus est une méthode couramment utilisée en animation pour simuler des objets déformables. Tout d'abord, un objet déformable est considéré comme une partie de l'espace bornée par une surface, qui définit l'intérieur et l'extérieur de l'objet. Ensuite, l'objet est soumis à un ensemble de forces extérieures

qui le déforment et le déplacent, comme la gravité, la force de frottement ou les forces appliquées par l'utilisateur.

Le calcul du déplacement de chaque point de l'objet par rapport à sa position de repos est la première étape dans la méthode de la mécanique des milieux continus. À partir de ces déplacements, nous pouvons déterminer le tenseur de déformation en tout point, qui décrit localement les déformations subies par l'objet. Ce tenseur est ensuite utilisé avec une loi de comportement pour calculer le tenseur de contrainte en tout point. Enfin, la densité énergétique est utilisée pour déterminer la densité de force en tout point de l'objet. Nous allons maintenant voir plus en détails ces différentes étapes.

Champs et gradient de déformation

Un objet déformable est défini par une position au repos x_i^0 pour tout sommet i de l'objet. Lorsque des forces sont appliquées sur l'objet, ses points sont déplacés vers une position x_i. Dans la suite, nous parlerons simplement de la position au repos globale x^0 de l'objet et de sa position courante x.

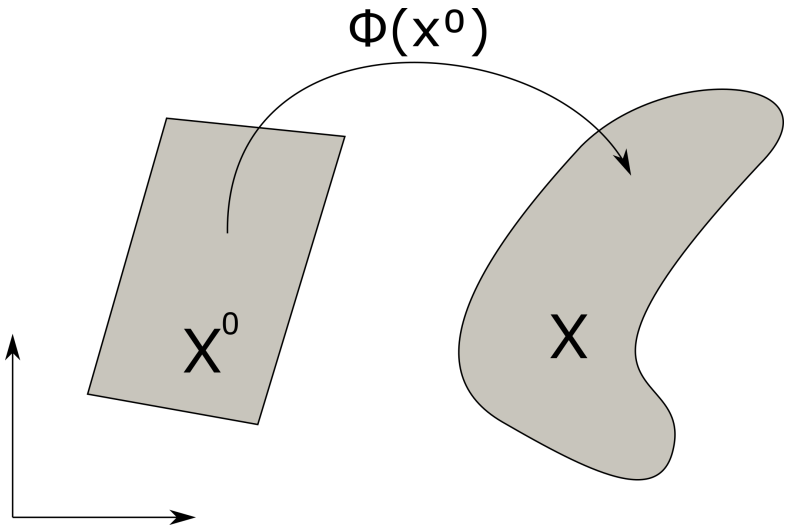

FIGURE 4.1 – Déformation d'un objet vers une configuration déformée

Nous pouvons ainsi définir une fonction $\phi(x^0) = x$ permettant de passer de la configuration initiale de l'objet à sa configuration déformée. Notre objectif est de déterminer dans quelle mesure notre objet est déformé afin de pouvoir le ramener à sa position initiale. Il est clair que lorsque les forces subies par l'objet n'induisent qu'une translation, l'objet ne subit pas de déformation entraînant une force interne. Une façon simple d'ignorer les translations est de prendre le gradient de déformation $F = \nabla\phi$. Il est également clair qu'une rotation n'entraîne pas de déformation sur l'objet. Par conséquent, nous avons besoin d'une mesure des déformations utilisant F qui soit indépendante des rotations.

Tenseur de déformation

Le tenseur de déformation est une mesure tensorielle qui décrit les changements de forme d'un objet lorsqu'il est soumis à une contrainte. L'une des propriétés principales de ce tenseur, noté ε, est qu'il doit être invariant par translation et rotation, ces transformations étant qualifiées de "rigides" et ne déformant pas l'objet. En 3D, le tenseur de déformation est représenté par une matrice 3×3 (voir équation 4.2).

$$\varepsilon = \begin{bmatrix} \varepsilon_x & \gamma_{xy} & \gamma_{xz} \\ \gamma_{yx} & \varepsilon_y & \gamma_{yz} \\ \gamma_{zx} & \gamma_{zy} & \varepsilon_z \end{bmatrix} \tag{4.2}$$

Les coefficients diagonaux ε_i représentent les élongations et compressions locales suivant les axes x, y ou z, et les autres coefficients γ_{ij} correspondent aux cisaillements. Il est important de noter que la matrice est symétrique et que donc $\gamma_{ij} = \gamma_{ji}$.

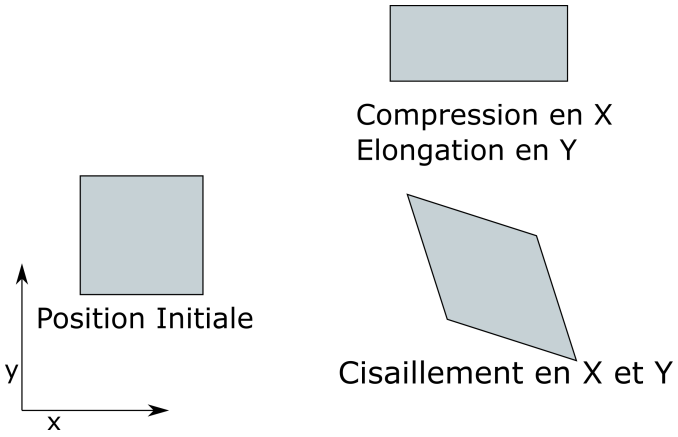

FIGURE 4.2 – Illustration des déformations capturées par le tenseur de déformation, les élongations/compressions et cisaillements suivant les différents axes

En informatique graphique, un choix populaire pour calculer le tenseur de déformation est le tenseur de Green-Lagrange, exprimé de la façon suivante (équation 4.3) :

$$\varepsilon = \frac{1}{2}(F^T F - I) \tag{4.3}$$

Dans cette expression, on utilise la propriété d'une matrice de rotation R qui est que $R^T R = I$; le terme $F^T F - I$ cherche donc à retirer la composante de rotation du gradient de déformation.

Pour de faibles déplacements, nous pouvons utiliser une version linéarisée de ce tenseur de déformation, le tenseur de Cauchy (équation 4.4) :

$$\varepsilon = \frac{1}{2}(F + F^T) - I \tag{4.4}$$

Ce tenseur a l'avantage d'être plus simple à obtenir que le tenseur de Green-Lagrange et, étant linéaire, de pouvoir être facilement interpolé. Néanmoins, ce tenseur n'est plus invariant par rotation, ce qui entraîne des déformations même lors de transformations rigides. Ce problème devra trouver des solutions dans les méthodes utilisant ce tenseur.

Tenseur de contrainte

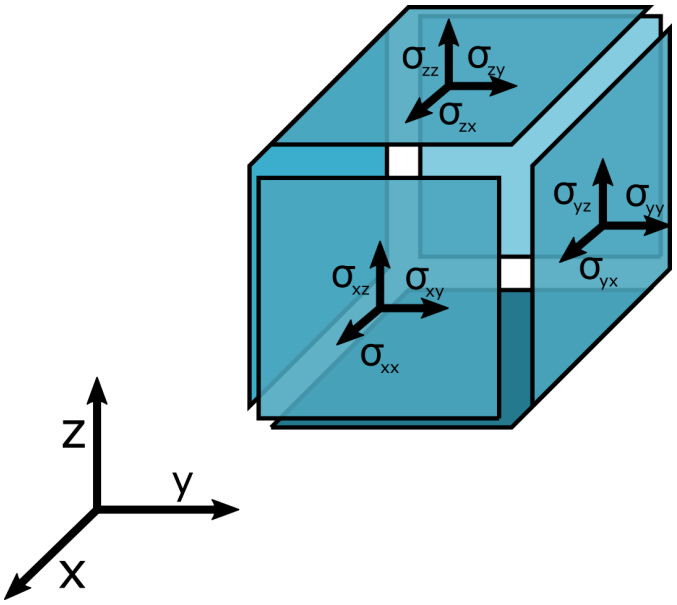

FIGURE 4.3 – Illustration du tenseur de contrainte.

Le tenseur de contrainte décrit les forces internes agissant sur un objet en réponse aux déformations locales, ces dernières étant décrites par le tenseur de déformation. Envisageons un cube infinitésimal centré autour de n'importe quel point de l'objet, dont les arêtes sont parallèles aux axes du repère (nous utiliserons un repère classique $\vec{\boldsymbol{e}}_{\mathbf{1}} = \vec{\boldsymbol{x}}$, $\vec{\boldsymbol{e}}_{\mathbf{2}} = \vec{\boldsymbol{y}}$ et $\vec{\boldsymbol{e}}_{\mathbf{3}} = \vec{\boldsymbol{z}}$). Chaque face de ce cube est numérotée, les faces i et $-i$ étant les faces ayant pour vecteur normal le vecteur $\vec{e}_i$. Nous pouvons ensuite définir une force pondérée par leur surface sur chacune de ces faces, définie par rapport au repère, ainsi σ_{ij} correspond à la composante du vecteur $\vec{F}$ dans la direction $\vec{e}_i$ sur la face ayant pour vecteur normal $\vec{e}_j$. Ce tenseur peut être représenté par une matrice 3×3 :

$$\sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} \tag{4.5}$$

Ce tenseur peut être calculé à l'aide d'une loi de constitution qui relie les déformations et les contraintes. Cette loi peut être déterminée de manière expérimentale ou théorique, en fonction des propriétés mécaniques de l'objet, telles que sa rigidité et son élasticité. Une loi de constitution est ainsi une fonction f du tenseur de déformation, il existe de

nombreuses lois de déformation dans la littérature, Dans le cadre de cette thèse, nous nous concentrerons sur les lois de comportement élastique, et en particulier sur la loi de comportement linéaire pour de petites déformations, connue sous le nom de loi de Hooke.

Loi de Hooke

La loi de Hooke, énoncée par le physicien Robert Hooke, est un principe fondamental en mécanique des matériaux. Elle vise à modéliser la relation entre le tenseur de contrainte et le tenseur de déformation d'un matériau isotrope (ayant les mêmes propriétés dans toutes les directions). Cette relation est approximée par une fonction linéaire :

$$\sigma = f(\varepsilon) = C\varepsilon \tag{4.6}$$

Ici, C est un tenseur de comportement d'ordre 4 possédant 81 coefficients. Cependant, pour un matériau isotrope, seulement deux coefficients dits de Lamé, λ et μ, sont nécessaires pour décrire le comportement élastique :

$$\sigma = \lambda tr(\varepsilon)I + 2\mu\varepsilon \tag{4.7}$$

où $tr(\varepsilon)$ représente la trace de la matrice ε (c'est-à-dire la somme des termes diagonaux de ε), et I est la matrice identité. Le coefficient λ n'a pas d'interprétation physique claire, tandis que le coefficient μ est le module de cisaillement, qui décrit la réaction de l'objet à une contrainte de cisaillement. Parfois, il est plus pratique de décrire le comportement d'un objet à l'aide de son module de Young E (qui correspond à la réponse d'un matériau à une contrainte uniaxiale dans la direction de cette contrainte) et de son coefficient de Poisson υ (qui décrit le comportement dans les directions perpendiculaires à cette contrainte). Les coefficients de Lamé peuvent être calculés à partir de E et υ à l'aide des relations suivantes :

$$\begin{aligned} \lambda &= \frac{E\nu}{(1+\nu)(1-2\nu)} \\ \mu &= \frac{E}{2(1+\nu)} \end{aligned} \tag{4.8}$$

Pour une représentation simplifiée, compte tenu de la symétrie des tenseurs de déformation et de contrainte, on peut les représenter par des vecteurs contenant leurs 6 coefficients significatifs :

$$\sigma = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{bmatrix} \varepsilon = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{bmatrix} \tag{4.9}$$

Ainsi, le tenseur de comportement est représenté par une matrice 6×6 :

$$C = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{4.10}$$

La loi de Hook est la principale loi de comportement utilisé en informatique graphique, même si le fait qu'elle ne soit correcte que pour de faible contrainte peut poser un problème dans certains contextes, pour cela d'autres lois peuvent être utilisées comme le modèle Néo-Hookéen.

Équation de Navier

Nous avons maintenant déterminé, à partir des déformations subies par l'objet, le tenseur de contrainte en tout point. Cependant, nous ne pouvons pas encore exploiter la mécanique du point pour animer le maillage représentant l'objet. Pour cela, nous utilisons l'équation de Navier, qui décrit le mouvement de chaque point de l'objet :

$$f_{ext} + f_{int} = \rho * \ddot{x} \tag{4.11}$$

Cette équation est similaire à la seconde loi de Newton vue précédemment, où ρ est la masse volumique du point et f_{int} sont les forces internes appliquées sur ce point obtenues à partir du gradient du champ de tenseurs de contrainte : $f_{int} = \nabla\sigma$.

4.1.4 Intégration temporelle

Nous avons maintenant vu les principes de base permettant de calculer les forces en tout point d'un maillage. Avant de nous intéresser aux algorithmes permettant de résoudre efficacement ces équations différentielles, nous allons voir comment passer de l'accélération d'une particule à sa position.

Nous avons vu précédemment que nous pouvons relier l'accélération d'une particule à la masse et aux forces appliquées sur cette particule. Notre objectif est maintenant de déterminer la vitesse et la position x de la particule pour un ensemble discret d'instant $x(0), x(\Delta t), x(2\Delta t)...$, avec Δt appelé pas de temps. La méthode la plus simple est le schéma d'intégration explicite qui consiste à remplacer les dérivées de notre système d'équations par des différences finies :

$$\begin{cases} \ddot{x}(t) = \frac{\dot{x}(t+\Delta t) - \dot{x}(t)}{\Delta t} \\ \dot{x}(t) = \frac{x(t+\Delta t) - x(t)}{\Delta t} \end{cases} \tag{4.12}$$

Que nous pouvons modifier en utilisant la définition précédente de l'accélération en :

$$\begin{cases} x(t+\Delta t) = x(t) + \Delta t \dot{x}(t) \\ \dot{x}(t+\Delta t) = \dot{x}(t) + \Delta t \frac{F(x(t), \dot{x}(t), t)}{m} \end{cases} \tag{4.13}$$

On obtient ainsi le schéma d'Euler explicite. Ce schéma d'intégration est dit explicite, car il donne explicitement les valeurs de la vitesse et de la position de la particule au prochain pas de temps. La qualité d'un schéma d'intégration est mesurée par sa précision et sa stabilité. La précision est mesurée par la convergence de la solution par rapport au pas de temps Δt en ordre de grandeur $(O(\Delta t), O(\Delta t^2), ...)$. L'instabilité se traduit par un éloignement non borné du résultat par rapport à la solution exacte. Souvent en informatique graphique, on considère que la stabilité est plus importante que la précision d'un schéma d'intégration.

Le schéma d'Euler explicite que nous venons de voir à une précision linéaire en fonction du pas de temps, mais n'est que conditionnellement stable, c'est-à-dire que la stabilité n'est atteinte que pour de faibles valeurs de Δt, ce qui peut poser des problèmes pour la simulation d'objets rigides. Par exemple, pour simuler un ressort accroché à une particule de masse m et de raideur k, Δt doit être inférieur à $2\sqrt{\frac{m}{k}}$, ce qui aboutit à de très faibles valeurs de Δt pour un coefficient de raideur important.

Pour résoudre ce problème, il existe des schémas d'intégration dits implicites, qui prennent en compte la dérivée au pas de temps suivant plus qu'au pas de temps actuel, comme le schéma d'Euler implicite :

$$\begin{cases} x(t+\Delta t) = x(t) + \Delta t \dot{x}(t+\Delta t) \\ \dot{x}(t+\Delta t) = \dot{x}(t) + \Delta t \frac{F(x(t+\Delta t), \dot{x}(t+\Delta t), t)}{m} \end{cases} \tag{4.14}$$

On remarque dans ces équations que les termes en $t+\Delta t$ sont présents des deux côtés de l'égalité, ce qui pose un problème pour la résolution du système qui va demander une méthode de résolution numérique comme la méthode de Newton-Raphson.

Cependant, bien que ces méthodes soient coûteuses et difficiles à mettre en place, les méthodes d'intégration implicites sont toujours stables, même si elles ont tendance à ajouter des frottements dans la simulation.

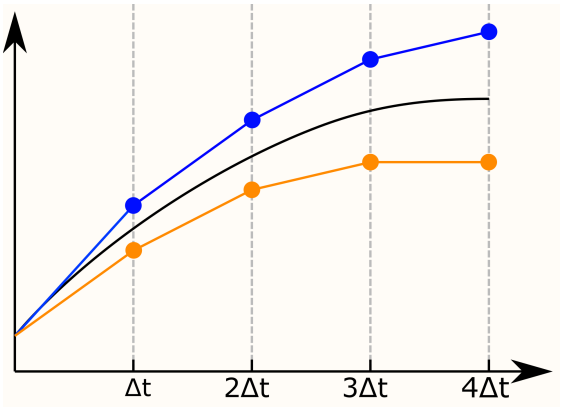

FIGURE 4.4 – Illustration des schémas d'Euler explicite (en bleu) et implicite (en orange) nous cherchons ici à approximer la courbe noire sur 4 pas de temps.

Il est important de noter qu'une amélioration simple de la méthode d'Euler explicite à mi-chemin de la méthode implicite est la méthode d'Euler symplectique :

$$\begin{cases} \dot{x}(t+\Delta t) = \dot{x}(t) + \Delta t \frac{F(x(t),\dot{x}(t),t)}{m} \\ x(t+\Delta t) = x(t) + \Delta t \dot{x}(t+\Delta t) \end{cases} \tag{4.15}$$

Cette méthode est aussi coûteuse, mais plus stable que la méthode d'Euler explicite. Le choix d'une méthode d'intégration dépend donc de plusieurs paramètres, tels que la précision des calculs, le temps alloué à la simulation ou l'espace mémoire utilisé.

Nous venons de voir les principes mécaniques de base permettant de mesurer les forces appliquées sur un objet déformable et comment passer de ces forces à la position de l'objet pour un ensemble discret d'instant dans le temps, pour plus de détails, nous invitons le lecteur à consulter [KE22]. Nous allons maintenant voir les principaux outils numériques nous permettant de calculer ces forces.

4.2 Modèles de comportement

En informatique graphique, l'une des problématiques majeures réside dans la représentation d'objets qui, par nature, sont discrets, comme cela a été exposé en détail dans les chapitres précédents concernant les maillages. Cependant, les modèles de comportement que nous utilisons pour animer ces objets reposent sur une modélisation continue, principalement grâce à l'utilisation fréquente d'opérateurs différentiels. L'objectif de cette section est de passer en revue les méthodes disponibles dans la littérature pour résoudre cette divergence conceptuelle. Nous nous pencherons ensuite plus en détail sur un ensemble restreint de techniques que nous avons mises en œuvre tout au long de cette thèse.

4.2.1 Introduction

La simulation d'objets déformables est une technique essentielle en informatique graphique qui permet d'animer des personnages, des vêtements et des objets du monde réel. Ces méthodes de simulation se divisent généralement en deux grandes catégories : les méthodes discrètes, qui approximent un objet continu en utilisant un ensemble de particules interagissantes, et les méthodes continues, qui reposent sur les équations de la mécanique des objets continus, résolues à l'aide de méthodes numériques pour simuler des objets déformables.

4.2.2 Méthodes discrètes

Dans le contexte des méthodes discrètes, chaque objet est discrétisé en un ensemble de points appelés particules. Chacune de ces particules est caractérisée par sa masse m_i, sa position p_i, ainsi que sa vitesse v_i. Ces particules interagissent entre elles par le biais de liaisons. Généralement, lorsqu'un objet est représenté par un maillage, les particules correspondent aux sommets du maillage, et les interactions entre particules sont représentées par les arêtes du maillage.

Masse ressort

Parmi ces méthodes, on trouve les systèmes masses-ressorts, qui sont des modèles discrets où la masse de l'objet est répartie sur les sommets de son maillage, et où les sommets sont reliés entre eux par différents types de ressorts.

Un ressort est généralement défini comme étant linéairement élastique et respectant une variante de la loi de Hooke en une dimension. Pour deux particules x_i et x_j reliées par un ressort de longueur au repos l_{ij}, la force f_i appliquée sur x_i s'écrit :

$$f_i = k(|x_{ij}| - l_{ij})\frac{x_{ij}}{|x_{ij}|} \tag{4.16}$$

où x_{ij} est le vecteur reliant x_i à x_j et k est le coefficient de rigidité du ressort.

Les systèmes masses-ressorts sont largement utilisés en informatique graphique et en simulation biomédicale, car ils sont simples, efficaces, faciles à implémenter et permettent d'obtenir des résultats visuellement satisfaisants. Il existe d'autres types de ressorts, tels que ceux agissant sur la vitesse de l'objet pour appliquer des frottements ou ceux agissant sur l'angle entre deux faces adjacentes.

Cependant, les systèmes masses-ressorts présentent des limites, notamment en termes de contrôle. Il est en effet difficile de trouver les bons paramètres de raideur pour représenter un comportement réel, même si des travaux ont été menés à ce sujet dans la littérature. De plus, l'utilisation de ressorts risque d'entraîner un effet d'anisotropie dans la simulation, étant donné que les ressorts n'appliquent une force que dans une direction.

4.2.3 Shape Matching

La méthode du *Shape Matching*, proposée par Müller et al. [Mül+05], est une approche populaire pour simuler les déformations d'objets déformables. Elle doit sa popularité à sa simplicité et à ses performances en temps réel. Dans cette section, nous décrivons le fonctionnement de cet algorithme et les concepts clés associés.

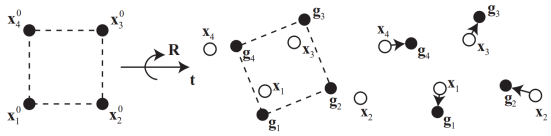

FIGURE 4.5 – Illustration du principe du *Shape Matching*, on cherche la rotation R et la translation t permettant de ramener les sommets dans la configuration initiale au plus proche de la configuration déformée. (Image issue de [Mül+05])

Le *Shape Matching* est appliqué à un objet discrétisé par un ensemble de points i. Chaque point possède une position courante x_i et une position initiale x_i^0. La position initiale correspond à la configuration non déformée de l'objet, tandis que la position courante représente la configuration de l'objet à un instant t après avoir subi des déformations. L'objectif du *Shape Matching* est de trouver la transformation rigide (matrice de rotation R et vecteurs de translation t et t^0) qui minimise (au sens des moindres

carrés) la différence entre la configuration initiale et la configuration courante de l'objet comme illustré dans la Figure 4.5.

La fonction à minimiser et :

$$\sum_i m_i(\mathbf{R}(x_i^0 - \mathbf{t}^0) + \mathbf{t} - x_i)^2 \tag{4.17}$$

où m_i représente la masse de chaque point. Les vecteurs de translation t et t^0 correspondent aux centres de masse des configurations initiale et courante, définis comme suit :

$$t_0 = x_{cm}^0 = \frac{\sum_i m_i x_i^0}{\sum_i m_i}, t = x_{cm} = \frac{\sum_i m_i x_i}{\sum_i m_i} \tag{4.18}$$

La détermination de la matrice de rotation R est moins directe. Nous définissons d'abord deux quantités $q_i = x_i^0 - t^0$ et $p_i = x_i - t$, puis reformulons le problème de recherche de R en cherchant une matrice de transformation A minimisant :

$$\sum_i m_i(Aq_i - p_i)^2 \tag{4.19}$$

En fixant la dérivée par rapport à tous les coefficients de A à zéro, nous trouvons la solution suivante :

$$A = (\sum_i m_i p_i q_i^T)\ (\sum_i m_i q_i q_i^T)^{-1} = A_{pq}A_{qq} \tag{4.20}$$

La matrice A_{qq} ne contenant pas de rotation, nous pouvons obtenir la matrice R en effectuant la décomposition polaire de la matrice $A_{pq} = RS$, où $S = \sqrt{A_{pq}^T A_{pq}}$ et $R = A_{pq}S^{-1}$. Une fois, la translation et la rotation optimales déterminées, un objectif peut être associé à chaque point :

$$g_i = R(x_i^0 - t^0) + t \tag{4.21}$$

Ces objectifs nous permettent de mettre à jour la vitesse et la position de tous les points de l'objet :

$$\begin{cases} v_i(t+h) &= v_i(t) + \alpha\frac{g_i(t) - x_i(t)}{h} + \frac{hf_{ext}(t)}{m_i} \\ x_i(t+h) &= x_i(t) + hv_i(t+h) \end{cases} \tag{4.22}$$

avec $\alpha \in [0, 1]$ un coefficient permettant de simuler la rigidité de l'objet. Plus α est proche de 1, plus l'objet est rigide.

Dans cette première approche du Shape Matching, seules les déformations mineures de la forme initiale sont prises en compte. L'article original de Muller et al. suggère deux méthodes principales pour représenter des déformations plus conséquentes.

D'une part, il est possible de réviser le calcul des objectifs g_i afin d'accroître le nombre de modes de déformation envisageables pour l'objet. D'autre part, une autre solution consiste à séparer l'ensemble des particules en plusieurs groupes, puis à appliquer le

Shape Matching individuellement à chacun d'eux. Ensuite, on calcule la moyenne des contributions de chaque groupe sur les particules. Lorsqu'un objet est représenté par un maillage volumique, ces groupes peuvent simplement correspondre aux éléments du maillage, tels que les tétraèdres, les hexaèdres, etc.

On retrouve différentes méthodes dans la littérature [SOG08 ; RJ07 ; DBB11] permettant d'utiliser des plus grands groupe de particule se superposant les uns aux autres, le principal challenge de ces méthodes est que lorsque la taille des groupes augmente le nombre d'additions nécessaires pour calculer la contribution de chaque sommet augmente de façon cubique en fonction de la taille des régions.

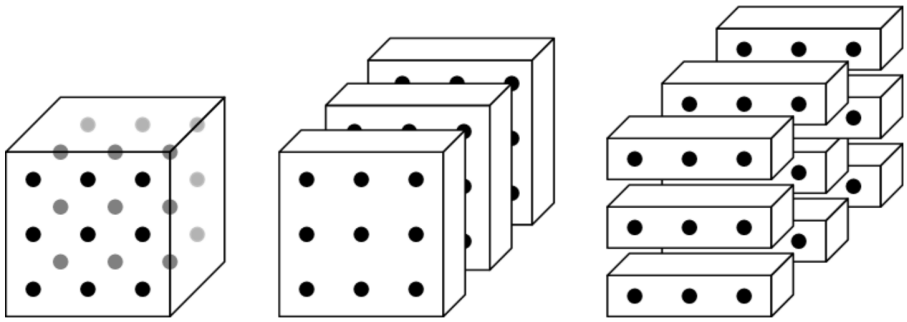

FIGURE 4.6 – Issue de [RJ07] décomposition en sous-sommation d'une région de taille 1

La méthode *Fast Lattice Shape Matching* [RJ07] est une approche qui applique le Shape Matching à une grille régulière dans laquelle l'objet à animer est plongé. Les sommets de la grille servent de degrés de liberté pour la simulation. Comme il s'agit d'une grille régulière, chaque sommet s_{xyz} se voit attribuer une coordonnée. On peut alors définir une sommation d'une quantité pour une région r de cette grille centrée sur le sommet s_{xyz} comme suit :

$$S_r = \sum_{k=z-w}^{z+w} \sum_{j=y-w}^{y+w} \sum_{i=x-w}^{x+w} s_{ijk} = \sum_{k=z-w}^{z+w} \left(\sum_{j=y-w}^{y+w} \left(\sum_{i=x-w}^{x+w} s_{ijk} \right) \right) \tag{4.23}$$

Cette décomposition permet de définir des sous-sommations, comme l'illustrent la figure 4.6 et l'équation 4.24 :

$$\underbrace{X_{xyz} = \sum_{i=x-w}^{x+w} v_{iyz}}_{\text{Somme sur les X (Barres)}} \Rightarrow \underbrace{XYxyz = \sum_{j=y-w}^{y+w} X_{xjz}}_{\text{Somme sur les Y (Plaques)}} \Rightarrow \underbrace{SUM_r = \sum_{k=z-w}^{z+w} XY_{xyk}}_{\text{Somme sur les Z (Cubes)}} \tag{4.24}$$

En utilisant ce découpage, il est possible de réaliser les sommations nécessaires au Shape Matching en $O(wn)$, où w représente la taille des régions et n le nombre total de sommets. En appliquant le schéma de récurrence illustré dans l'équation 4.25, on obtient une complexité en temps constant par rapport à la taille des régions.

$$\begin{cases} X_{xyz} & = & X_{(x-1)yz} - v_{(x-w-1)yz} + v_{(x+1)yz} \\ XY_{xyz} & = & XY_{x(y-1)z} - X_{x(y-w-1)z} + X_{x(y+1)z} \\ SUM_{xyz} & = & SUM_{xy(z-1)} - XY_{xy(z-w-1)} + XY_{xy(z+1)} \end{cases} \tag{4.25}$$

La méthode du *Fast Lattice Shape Matching* permet donc un calcul rapide du shape matching par région, mais n'est utilisable que sur des grilles régulières. Diziol et al. proposent une méthode similaire, utilisable sur tout maillage surfacique triangulaire, reposant sur le découpage des régions (2D) en ligne pouvant être calculées efficacement, permettant ainsi une complexité linéaire en fonction de la taille des régions.

La méthode du Shape Matching présente plusieurs inconvénients. Tout d'abord, il s'agit d'une méthode d'animation discrète qui ne repose pas sur la physique, ce qui peut poser des problèmes, notamment dans le cadre d'une simulation adaptative où les zones plus fines deviennent progressivement moins rigides. Deuxièmement, dans sa formulation actuelle, le comportement de l'objet est fortement lié à la taille du pas de temps choisi : un pas de temps important entraîne un comportement plus rigide que celui d'un pas de temps plus faible.

Pour pallier ces défauts, deux améliorations ont été proposées dans la littérature. La première, la *dynamique basée sur la position étendue*, permet de rendre la simulation indépendante de la taille du pas de temps choisi. La seconde amélioration, le *Shape Matching basé sur la physique* [Mül+22], utilise les lois de la mécanique des milieux continus pour animer l'objet. Nous explorons ces deux améliorations ci-dessous.

4.2.4 Amélioration de la méthode du Shape Matching basée sur la physique

Le Shape Matching s'inscrit dans le cadre de la dynamique basée sur la position (PBD) [Mül+07] où les différentes forces sont exprimées sous forme de contraintes. Dans sa forme de base, la rigidité des contraintes dépend du pas de temps choisi pour la simulation. La contribution de l'XPBD [MMC16] est de proposer un multiplicateur de Lagrange λ défini à partir de la rigidité du modèle α :

$$\lambda = \frac{-C(\boldsymbol{x})}{\sum_{i=1}^{n} w_i |\nabla_{x_i} C(\boldsymbol{x})|^2 + \frac{\alpha}{h^2}} \tag{4.26}$$

La position x_i d'une particule est ensuite définie comme :

$$x_i = x_i + \lambda w_i \nabla_{x_i} C(\boldsymbol{x}) \tag{4.27}$$

La contribution du Shape Matching basé sur la physique [Mül+22] est de proposer une formulation des contraintes et du facteur de rigidité α basée sur le modèle néo-hookean. Pour cela, deux contraintes sont définies :

$$C_H(F) \quad = \quad det(F) - \gamma \; C_D(F) \quad = \quad \sqrt{tr(F^T F)} \tag{4.28}$$

Avec F le gradient de déformation défini de la même façon que pour le Shape Matching classique :

$$F = (\sum_i m_i p_i q_i^T)\ (\sum_i m_i q_i q_i^T)^{-1} = PQ^{-1} \tag{4.29}$$

Ces deux contraintes sont associées à deux paramètres de rigidité :

$$\alpha_H \quad = \quad \frac{1}{\lambda V_e}\ \alpha_D \quad = \quad \frac{1}{\mu V_e} \tag{4.30}$$

avec λ et μ les paramètres de Lamé et V_e le volume courant de l'élément. Les gradients des deux contraintes sont ensuite exprimés comme suit, avec $F = \begin{bmatrix} f_1 & f_2 & f_3 \end{bmatrix}$:

$$\begin{array}{rcl} \nabla_{x_i} C_H(\boldsymbol{x}) & = & m_i[f_2 \wedge f_3, f_3 \wedge f_1, f_1 \wedge f_2]Q^{-T}q_i \\ \nabla_{x_i} C_D(\boldsymbol{x}) & = & \frac{m_i}{\sqrt{|f_1|^2+|f_2|^2+|f_3|^2}} FQ^{-T}q_i \end{array} \tag{4.31}$$

Appliquer ces contraintes sur l'ensemble des volumes de l'objet permet d'obtenir un comportement réaliste. Cependant, le gradient de déformation exprimé sous cette forme ne prend pas en compte tous les degrés de liberté d'un volume ayant plus de quatre sommets. Ce phénomène peut entraîner des déplacements de l'objet non pris en compte par les contraintes, de manière similaire à l'effet de sablier observé dans la méthode des éléments finis. Pour pallier ce problème, Müller et al. proposent d'ajouter une contrainte supplémentaire pour corriger la position des particules :

$$x_i = x_{\text{cm}} + Fq_i \tag{4.32}$$

Le Shape Matching basé sur la physique permet ainsi d'obtenir un comportement réaliste et contrôlable des objets déformables, applicable à tout type d'élément volumique (tétraèdre, hexaèdre, etc.) tout en offrant une grande simplicité de mise en œuvre.

4.2.5 Méthodes Continue

Nous allons maintenant nous intéresser aux méthodes continues, contrairement aux méthodes discrètes les objet sont ici considérés comme une couverture continue de l'espace délimité par une surface, le principal challenge de ces méthodes sera ainsi de pouvoir représenter les équations de la mécanique des milieux continus à l'aide de l'informatique qui est intrinsèquement discret. Différentes méthodes numériques ont été proposées pour implanter la mécanique des milieux continus. Nous pouvons citer la méthode des différences finies et celle des éléments finis. La méthode des différences finies cherche à approximer les opérateurs différentiels utilisés dans les définitions du gradient de déformation ou dans l'équation de Navier. En utilisant l'approximation suivante pour la dérivée des déplacements :

$$\frac{\partial u}{\partial x_i}(x) \approx \frac{u(x + \delta_i x_i) - u(x - \delta_i x_i))}{2\delta_i} \tag{4.33}$$

En pratique, avec la méthode des différences finies, chaque objet est modélisé sous forme d'une grille régulière. Dans la formule précédente, x représente un nœud de la grille et $x + \delta_i x_i$ et $x - \delta_i x_i$ ses voisins dans la direction x_i. Toutefois, la version des différences

finies proposée par Terzopoulos et al. présente un inconvénient majeur, à savoir qu'elle n'est applicable que sur une grille régulière avec des points disposés avec un écart de δ_i le long de chaque axe, ce qui ne permet pas de représenter des topologies fines et encore moins d'être utilisable dans un contexte adaptatif.

Debunne et al. [Deb+] ont proposé une extension de cette méthode, utilisable sur tout type de maillage. Néanmoins, la méthode des différences finies reste trop instable et pose des problèmes pour les nœuds situés sur les bords de l'objet, qui disposent d'un voisinage incomplet.

La méthode des éléments finis (FEM) est une approche courante pour résoudre des problèmes de mécanique des milieux continus. Elle repose sur la discrétisation des objets en un ensemble d'éléments finis, tels que des tétraèdres ou des hexaèdres en trois dimensions. Cette méthode est utilisée pour estimer les propriétés physiques à l'intérieur de chaque élément en utilisant une interpolation basée sur les valeurs connues aux sommets ou nœuds de l'élément. Les fonctions d'interpolation, également appelées fonctions de forme, sont définies pour chaque type d'élément et peuvent être linéaires ou quadratiques, selon les besoins.

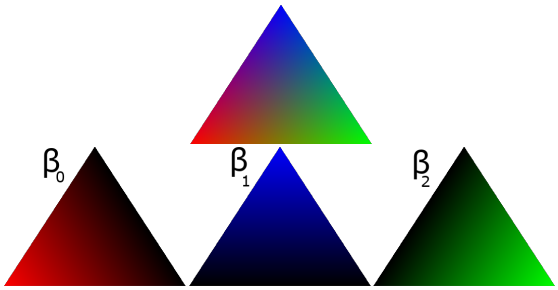

FIGURE 4.7 – Illustration des fonctions de forme pour un triangle en 2D

Prenons l'exemple d'un maillage tétraédrique où les fonctions de forme sont basées sur les coordonnées barycentriques, notées $b = [u, v, w]^T$, à l'intérieur de chaque tétraèdre, comme illustré dans la Figure 4.7. Les fonctions de forme pour ce maillage sont les suivantes :

$$\begin{aligned}\beta_0(b) &= 1 - u - v - w \\ \beta_1(b) &= u \\ \beta_2(b) &= v \\ \beta_3(b) &= w\end{aligned}$$

La position des points à l'intérieur du tétraèdre est calculée en fonction de la position de ses nœuds à l'aide de la relation suivante : $x(b) = \sum_{i=0}^{3} x_i \beta_i(b)$.

En utilisant ces fonctions de forme, nous pouvons résoudre différentes équations différentielles impliquées dans la mécanique des milieux continus, telles que le calcul des

gradients de déformation F pour chaque tétraèdre :

$$F = \frac{\partial x}{\partial x^0} = \frac{\partial x}{\partial b}\frac{\partial b}{\partial x^0} = \frac{\partial x}{\partial b}(\frac{\partial x^0}{\partial b})^{-1} = \frac{\partial}{\partial b}(\sum_{i=0}^{3} x_i\beta_i(b))(\frac{\partial}{\partial b}\sum_{i=0}^{3} x_i^0\beta_i(b))^{-1}$$

Cependant, la méthode des éléments finis ne se limite pas à la résolution de problèmes statiques. Il existe des méthodes dites *statiques* où une matrice de rigidité K est construite à partir de l'ensemble des éléments de l'objet et de leurs fonctions de forme. Le problème consiste ensuite à trouver la matrice de déplacement des nœuds U pour résoudre l'équation $R = KU$, où R représente les forces externes telles que des contacts ou la gravité. Cependant, cette approche est généralement coûteuse en termes de temps de calcul et n'est pas adaptée à la simulation en temps réel.

Une approche plus préférable est d'utiliser une formulation *explicite* des éléments finis, basée sur la deuxième loi de la dynamique exprimée en chaque nœud :

$$m_i\ddot{\mathbf{u}}_i + \lambda_i\dot{\mathbf{u}}_i - \mathbf{f}_{\text{int},i} = \mathbf{f}_{\text{ext},i}$$

où m_i est la masse du nœud i, λ_i est le coefficient de viscosité, $\mathbf{f}_{\text{int},i}$ est la somme des forces issues des déformations des éléments auxquels le nœud appartient, et $\mathbf{f}_{\text{ext},i}$ est la somme des forces extérieures.

La méthode des éléments finis est largement utilisée pour résoudre des équations basées sur les lois de la mécanique continue, que ce soit en ingénierie mécanique ou en informatique graphique. Nous allons maintenant nous intéresser plus particulièrement à trois méthodes de simulation que nous avons mises en place durant cette thèse.

4.2.6 Smoothed Particle Hydrodynamics (SPH)

La méthode numérique *Smoothed Particle Hydrodynamics* (SPH) [Mon92] est largement utilisée en mécanique des fluides et des solides déformables pour simuler des matériaux en utilisant des particules munies d'une fonction noyau. Dans cette section, nous présenterons les principes fondamentaux de cette méthode et les étapes nécessaires pour son implantation.

Principe

Le principe de la méthode SPH est d'estimer la valeur $A(x)$ d'une fonction A et de son gradient $\nabla A(x)$ pour chaque particule x. Pour cela, on utilise le fait que :

$$A(x) = (A * \delta)(x) = \int A(x')\delta(x - x')dx' \tag{4.34}$$

où $*$ est l'opérateur de convolution et $\delta(r)$ la fonction de Dirac définie par :

$$\delta(r) = \begin{cases} \infty & \textbf{si } r = 0 \\ 0 & \textbf{sinon} \end{cases} \qquad \textbf{et} \qquad \int \delta(r)dr = 1 \tag{4.35}$$

Le principe des SPH est d'approcher la fonction Dirac par une fonction noyau $W(x,h)$, avec h correspondant à la longueur de lissage de la fonction noyau, ce qui permet de lisser la fonction A et d'utiliser la propriété suivante :

$$A(x) = (A * \delta)(x) \approx (A * W)(x) = \int A(x')W(x - x', h)dv' \tag{4.36}$$

La fonction $W(x,h)$ doit respecter un certain nombre de propriétés :

$$\int W(r', h)dr' = 1$$

$$\lim_{h \to 0} W(r,h) = \delta(r)$$

$$W(r,h) \geq 0$$

$$W(r,h) = W(-r,h)$$

$$W(r,h) = 0 \; if \; ||r|| \geq h$$

Dans le cadre de la simulation physique, il est également souhaitable que la fonction noyau soit deux fois continûment dérivable. Le rayon r définit l'aire d'influence d'une particule comme illustré dans la Figure 4.8.

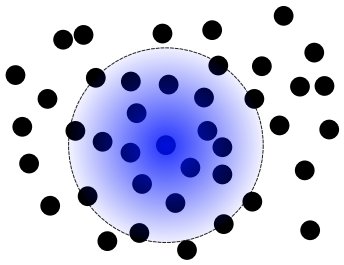

FIGURE 4.8 – Illustration de l'aire d'influence d'une particule

La dernière étape de la méthode SPH est de discrétiser l'intégrale en évaluant la fonction A pour un nombre fini de particules x_i échantillonnant l'espace :

$$(A * W)(x_i) = \int A(x')W(x_i - x', h_i)dx' \approx \sum_{x_j \in N_i} A(x_j)V_j W(x_i - x_j, h_i) \tag{4.37}$$

Avec N_i correspondant au voisinage de la particule x_i et V_j correspondant au volume associé à la particule x_j. Nous pouvons également approximer le gradient ∇A de A à l'aide des SPH comme suit :

$$\nabla A \approx \sum_{x_j \in N_i} A(x_j)V_j \nabla W(x_i - x_j, h_i) \tag{4.38}$$

La méthode SPH repose sur l'approximation de la fonction Dirac par une fonction noyau et la discrétisation des intégrales pour estimer les valeurs et les gradients des fonctions associées aux particules. La sélection d'une fonction noyau appropriée et la détermination de la longueur de lissage h sont cruciales pour la précision et la stabilité de la simulation. Il existe plusieurs fonctions noyau couramment utilisées dans les applications SPH, telles que le noyau de Wendland, le noyau gaussien et le noyau cubique spline, présentant des avantages et des inconvénients en termes de performance et de précision. Le choix de la fonction noyau dépend donc des applications.

Une fois la fonction noyau sélectionnée et les équations discrétisées, la méthode SPH peut être utilisée pour résoudre divers problèmes en mécanique des fluides et des solides déformables, tels que l'écoulement de fluides incompressibles et compressibles, la fragmentation, l'élasticité et la plasticité des solides, et les interactions fluide-structure. Pour plus d'informations sur les méthodes SPH, nous encourageons le lecteur à consulter l'état de l'art de Koschier et al. [Kos+22].

Simulation d'objet déformable

Après avoir présenté les principes généraux de la méthode SPH, nous allons examiner comment l'appliquer à la simulation d'objets déformables en suivant la méthode proposée par Peer et al. [Pee+18]. Dans cette approche, un objet est discrétisé en un ensemble de particules, chacune étant associée à trois quantités clés : une position courante x, une position initiale x^0 et un volume initial V^0. Plus généralement, toute quantité dans la configuration initiale de l'objet sera notée avec un $*^0$.

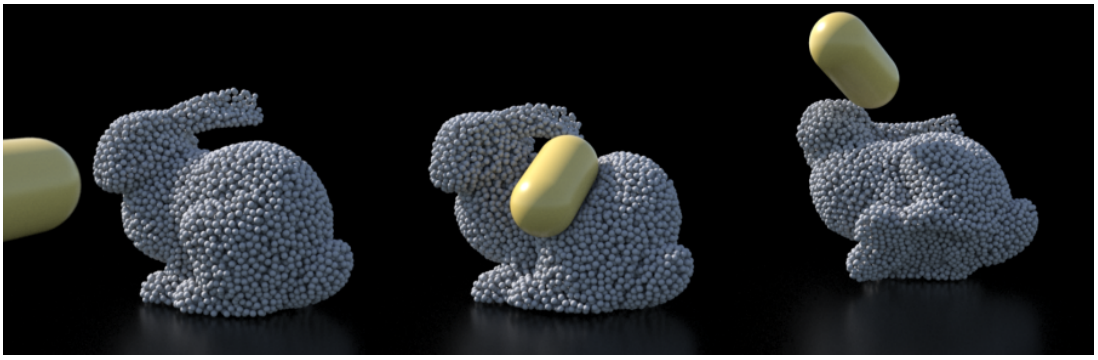

FIGURE 4.9 – Simulation d'un lapin déformable en interaction avec une capsule avec la méthode de Peer et al. (issu de [Pee+18])

Le gradient de déformation $F = \frac{\partial x}{\partial x^0}$ permet de décrire l'évolution des déformations de l'objet dans l'espace. Pour les matériaux étudiés, le tenseur de contrainte, utilisé pour calculer les forces élastiques, est déterminé uniquement par F. Le voisinage des particules, dans la configuration initiale, est déterminé en utilisant la topologie du maillage. On note $\sum_j^0$ pour parcourir les particules x_j dans le voisinage de x_i.

Une formulation du gradient avec les SPH est donnée par : $F_i = \sum_j^0 V_j^0(x_{ji}) \otimes \nabla W(x_{ij}^0, h_i)$, où $x_{ij} = x_i - x_j$. Cependant, cette formulation traite les rotations comme des déformations. Pour résoudre ce problème, Bonnet et Lok [BL99] proposent une matrice de correction :

$$L_i = \left(\sum_j^0 V_j^0 \nabla W(x_{ij}^0, h_i) \otimes (x_{ji}^0) \right)^{-1}$$

Ils calculent ensuite le gradient corrigé de la fonction noyau :

$$\nabla \tilde{W}(x_{ij}^0, h_i) = L_i \nabla W(x_{ij}^0, h_i)$$

Le gradient de déformation est alors calculé comme suit :

$$F_i = \sum_j^0 V_j^0(x_{ji}) \otimes \nabla \tilde{W}(x_{ij}^0, h_i)$$

À partir de ce gradient de déformation, le tenseur de déformation $\varepsilon = \frac{1}{2}\left(F_i + F_i^T\right) - I$ peut être obtenu. Comme ce tenseur est linéaire, il est important de retirer d'abord la composante de rotation R_i des déformations, responsable de la non-linéarité. Un nouveau gradient de la fonction noyau est ensuite défini :

$$\overset{*}{\nabla} W(x_{ij}^0) = R_i L_i \nabla W(x_{ij}^0, h_i)$$

Enfin, le gradient de déformation corrigé est calculé :

$$\overset{*}{F}_i = I + \sum_j^0 V_j^0 (x_{ji} - R_i x_{ji}^0) \otimes \overset{*}{\nabla} W(x_{ij}^0, h_i)$$

À partir de ce gradient corrigé, le tenseur de déformation peut être calculé :

$$\epsilon_i = \frac{1}{2}\left(\overset{*}{F}_i + \overset{*}{F}_i^T \right) - I$$

Ensuite, le tenseur de contrainte σ peut être calculé :

$$\sigma = \lambda tr(\varepsilon) I + 2\mu\varepsilon$$

avec λ et μ les coefficients de Lamé, la force élastique en chaque sommet peut être calculée comme la divergence de σ :

$$f_i = \sum_j^0 V_i^0 V_j^0 \left(\sigma_i \overset{*}{\nabla} W(x_{ij}^0, h_i) - \sigma_j \overset{*}{\nabla} W(x_{ji}^0, h_j) \right)$$

Cette méthode permet de simuler avec précision des objets déformables en utilisant la méthode SPH. Les forces élastiques sont calculées en tenant compte des déformations

et des rotations, offrant une représentation réaliste du comportement des objets déformables. La méthode de Peer et al. [Pee+18] est particulièrement adaptée à la simulation de matériaux élastiques, mais peut également être étendue à d'autres types de matériaux en ajustant les paramètres et les modèles constitutifs.

Au cours de cette section, nous avons présenté les fondements mécaniques permettant d'animer un objet déformable en informatique graphique. De plus, nous avons examiné en détail une sélection de méthodes visant à résoudre ces systèmes mécaniques. Dans la suite, nous nous intéresserons aux méthodes de simulation adaptative. Ces méthodes permettent de concentrer les degrés de liberté de la simulation dans les zones de plus grand intérêt afin d'optimiser les coûts en temps de calcul lors de la simulation d'objets complexes.

4.3 Adaptation des modèles

La simulation réaliste d'objets et de phénomènes complexes requiert l'utilisation de modèles de simulation avancés et un nombre important de degrés de liberté pour représenter avec précision ces phénomènes. Cependant, l'augmentation de la complexité des modèles et du nombre de degrés de liberté entraîne une augmentation correspondante du temps de calcul et de l'espace mémoire nécessaire. Pour résoudre ce problème, les méthodes adaptatives ont été développées.

Une méthode est qualifiée d'adaptative si elle est capable d'ajuster le modèle mécanique, les structures de données ou le maillage de l'objet animé en fonction de critères spécifiques. Ces critères peuvent inclure la garantie d'un temps de calcul compatible avec les contraintes de temps réel tout en minimisant l'erreur globale de la simulation, ou l'amélioration de la qualité de la simulation dans les zones d'interaction utilisateur.

Dans cette partie, nous présenterons un aperçu des méthodes de simulation adaptatives en commençant par les méthodes d'adaptation temporelle, puis en abordant les méthodes d'adaptation géométrique.

4.3.1 Adaptation temporelle

Lors de l'animation d'un objet simulé, l'intégration temporelle des équations de mouvement est essentielle pour assurer l'exactitude, la cohérence et la stabilité du système. Cependant, cette intégration nécessite parfois une adaptation en fonction des circonstances de la simulation.

Plusieurs raisons justifient cette adaptation temporelle. Par exemple, lorsqu'un système entre dans un régime fortement non linéaire, comme une fracture, des pas de temps plus petits sont nécessaires pour maintenir la précision souhaitée. De plus, l'ajustement du pas de temps peut être nécessaire pour éviter des états invalides, tels que des configurations d'interpénétration lors de collisions.

La stabilité des simulations mécaniques dépend de la relation entre le pas de temps, la résolution spatiale et la vitesse de propagation des informations dans le système. Ainsi,

si l'un de ces paramètres change en raison d'une adaptation spatiale ou d'un écoulement rapide, le pas de temps doit être ajusté pour maintenir la stabilité.

Il existe deux catégories principales de techniques d'adaptation temporelle. La première concerne la résolution temporelle, c'est-à-dire le choix approprié de la longueur du pas de temps en fonction des caractéristiques spécifiques de la simulation. La seconde catégorie concerne les techniques d'intégration elles-mêmes, en adaptant les méthodes d'intégration en fonction du contexte local.

Dans la première catégorie, l'adaptation des pas de temps peut se faire de manière globale en choisissant le plus petit pas de temps garantissant la stabilité sur l'ensemble de la simulation. Cependant, il est important de noter que l'utilisation de pas de temps adaptatifs n'est pas compatible avec tous les schémas d'intégration temporelle. Des travaux, tels que le schéma d'intégration mixte implicite/explicite proposé par Bridson et al., ont été développés pour garantir la stabilité lors de l'utilisation de pas de temps adaptatifs [BMF05].

Une autre approche consiste à utiliser un pas de temps adaptatif local, où chaque particule ou élément utilise un pas de temps différent. Par exemple, pour un ensemble de deux particules avec des positions respectives $x_i(t)$, les positions de ces particules au pas de temps suivant peuvent être calculées avec l'équation :

$$x_i(t + \Delta t) = x_i(t) + \Delta t f(x_0(t), x_1(t))$$

Dans cette approche, chaque particule a un pas de temps différent, noté Δt_i, avec Δt_0 inférieur à Δt_1. Deux approches peuvent être utilisées pour obtenir les positions $x_1(t + \Delta t_0)$, $x_1(t + 2\Delta t_0)$, etc.

La première approche consiste à interpoler la position de la particule entre $x_1(t)$ et $x_1(t+\Delta t_1)$. La seconde approche consiste à considérer que la fonction $f(x_0(t), x_1(t))$ est constante jusqu'à ce qu'un pas de temps Δt_1 s'écoule. Les deux approches permettent de réduire le coût d'évaluation de la fonction $f(x_0(t), x_1(t))$, qui est souvent la partie la plus coûteuse lors de l'intégration. Cette méthode d'adaptation locale du pas de temps est notamment utilisée par Goswami et al. et Fang et al. [GB14 ; Fan+18] pour simuler des fluides en divisant l'espace en blocs et en associant à chaque bloc un pas de temps unique.

En plus de ces techniques, d'autres méthodes d'adaptation temporelle existent, telles que l'adaptation de la méthode d'intégration elle-même. Certaines méthodes combinent des schémas d'intégration explicite et implicite pour traiter les déformations importantes de manière stable, comme la méthode IMEX de Fierz et al. [FSH11]. Une autre approche consiste à *geler* un sous-ensemble des degrés de liberté de l'objet, qui ne seront plus mis à jour pendant un certain temps. Cette méthode est particulièrement utile lorsque la zone simulée est vaste et que les interactions sont localisées dans l'espace.

En conclusion, l'adaptation temporelle est cruciale pour garantir la stabilité et l'efficacité des simulations physiques. En ajustant les pas de temps en fonction des caractéristiques de la simulation et en adaptant les méthodes d'intégration, il est possible d'obtenir des résultats plus précis et de résoudre des problèmes spécifiques de manière plus efficace.

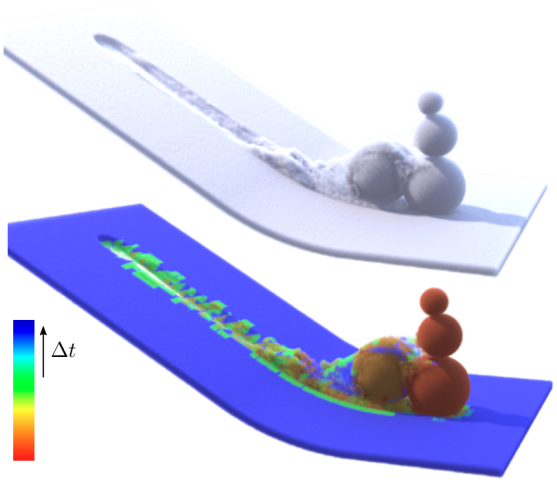

FIGURE 4.10 – Illustration de l'adaptation du pas de temps pour une simulation de neige utilisant les SPH, le bleu correspond aux pas de temps les plus élevé et le rouge aux pas de temps les plus faible (issu de [Fan+18])

4.3.2 Adaptation spatiale

La deuxième forme d'adaptation est l'adaptation géométrique c'est à dire le raffinement ou la simplification de la discrétisation de l'objet. Cette adaptation se décompose en deux étapes : premièrement évaluer un critère de raffinement permettant de déterminer où ajouter ou retirer des degrés de liberté et, deuxièmement, appliquer un schéma de raffinement permettant d'adapter la discrétisation pour résoudre le critère.

Ce type d'adaptation est la contribution principale de cette thèse, nous allons commencer par voir les différents critères de raffinement utilisés dans l'état de l'art avant de nous intéresser plus particulièrement aux différents types d'adaptations spatiales.

Critères de raffinement

Dans la littérature, différents critères de raffinement sont proposés. De nombreuses méthodes reposent sur des heuristiques simples, telles que la présence de contacts, la courbure de l'objet ou encore la distance par rapport au bord. Il est important de souligner que ces heuristiques ne sont pas dépourvues de fondement. En effet, en présence d'une forte courbure, des comportements visuels fortement non linéaires peuvent se produire et doivent être gérés avec précaution. De plus, il est également possible d'adapter le maillage en fonction du point de vue de la caméra, comme le montrent les travaux de Koh et al. pour la simulation de vêtements et la méthode de Servin et al. pour la simulation de câbles.

D'autres critères de raffinement reposent sur les propriétés physiques et géométriques de l'objet simulé. Wu et al. [Wu+01] répertorient plusieurs de ces critères. Les deux

premières méthodes se basent respectivement sur la concentration de contrainte et le champ de déplacement de l'objet, mesurant l'élongation relative des arêtes. Ces critères présentent l'avantage majeur d'être rapides à calculer. Dans leur méthode, l'estimation basée sur les contraintes représente environ 6% du temps de calcul d'un pas de temps, tandis que l'estimation basée sur l'élongation représente environ 10% du temps de calcul. Cependant, ces critères d'erreur ne sont pas optimaux et peuvent conduire à l'ajout de degrés de liberté dans des zones auxquelles cela n'est d'aucune utilité, notamment dans des zones dans lesquelles les déformations sont constantes, comme par exemple un cube écrasé dans toutes les directions.

Pour résoudre ce problème, Wu et al. proposent deux critères plus élaborés. Le premier consiste à simuler l'objet en utilisant des fonctions de forme de degré supérieur et à comparer les résultats avec la simulation actuelle. Le deuxième critère repose sur le gradient de contrainte, ce qui permet d'ignorer les zones où l'ajout de degrés de liberté aurait un impact limité. Ces deux méthodes nécessitent environ 16% du temps de calcul d'un pas de temps pour être évaluées.

Des valeurs non-scalaires peuvent également être utilisées comme critère de raffinement. Par exemple, Wicke et al. proposent de construire une matrice $\mathbf{M}$ de telle façon que les valeurs propres les plus importantes soient dans les directions où les déformations changent le plus rapidement. Cela permet de gérer le raffinement anisotropique du maillage.

Otaduy et al. proposent un critère basé sur leur méthode de simulation multigrille en comparant la position x^j d'une particule avec $P^j x^{j+1}$, la position obtenue depuis le précédent niveau plus grossier. Cette mesure est naturelle car elle évalue directement l'impact qu'aura l'ajout d'un degré de liberté sur la simulation. Ils proposent également de pondérer l'erreur par la matrice locale du système A afin d'éviter les possibles oscillations du système, comme suit :

$$e = ||A(x^j - P^j x^{j+1})||$$

En conclusion, de nombreux critères de raffinement sont disponibles dans la littérature, offrant différentes approches pour améliorer la discrétisation de l'objet simulé. Chaque critère présente ses avantages et ses limites, et leur choix dépendra du contexte spécifique de la simulation. Les travaux de Wu et al., Wicke et al. et Otaduy et al. [Wu+01 ; Ota+07 ; Wic+10] fournissent des méthodes plus élaborées pour surmonter les limitations des critères traditionnels. En intégrant ces critères dans les simulations physiques, il est possible d'obtenir des résultats plus précis et réalistes.

Adaptativité sur des maillages structurés

Une première catégorie de méthodes d'adaptation géométrique concerne les méthodes exploitant la symétrie ou la structure des modèles en utilisant des quadtrees en 2D ou des octrees en 3D. Par exemple, Hutchinson et al. [HPH96] ainsi que Villard et Borouchaki [VB05] utilisent des quadtrees pour connecter directement les masses et les ressorts, introduisant des jonctions en T dans le processus. Cette approche offre une

grande flexibilité pour la modélisation des systèmes masse-ressort, mais la convergence des masses-ressorts lors du raffinement n'est pas aussi clairement établie que dans les méthodes basées sur les éléments finis. Des études supplémentaires seraient nécessaires pour mieux comprendre et évaluer la convergence des masses-ressorts dans le contexte des méthodes utilisant des quadtrees ou des octrees.

En ce qui concerne la simulation d'objets élastiques, des méthodes basées sur les octrees ont également été développées . Ganovelli et al. [Gan99] proposent une méthode similaire en 3D basée sur des octrees et un système masse-ressort. Debunne et al. [Deb+99] simulent des modèles élastiques en utilisant une méthode approximant des différences finies, et ils utilisent un raffinement basé sur les octrees pour augmenter les détails. La Figure 4.11 illustre ces approches avec la méthode de Nesme et al. [NFP06]. Ces approches permettent d'obtenir des simulations précises d'objets élastiques, mais la convergence et la stabilité des modèles mécaniques utilisés peuvent poser des défis particuliers.

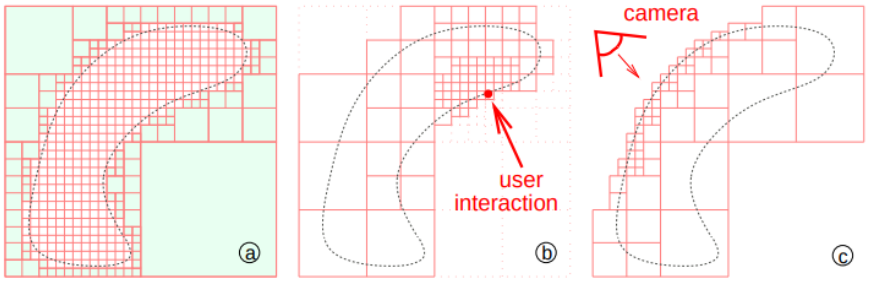

FIGURE 4.11 – Illustration de l'adaptation d'un octree selon les interactions utilisateurs ou le point de vue de la caméra (issu de [NFP06])

Il est important de noter que la discrétisation des équations aux dérivées partielles à l'aide de quadtrees et d'octrees peut présenter des effets secondaires subtils. Certaines propriétés utiles des méthodes de discrétisation régulière peuvent ne plus être préservées en présence de jonctions en T lorsqu'on utilise des octrees. Des ajustements spécifiques sont alors nécessaires pour préserver ces propriétés souhaitables dans le contexte des simulations d'objets élastiques.

Une autre méthode utilisant une structure en arbre exploite l'utilisation d'*adaptive BCC lattices*. Elle consiste, dans le cas régulier, à partir d'une grille régulière sur laquelle des sommets sont ajoutés sur les coins de la grille et au centre des cellules de la grille, puis une tétraédrisation de l'ensemble est réalisée. Dans le cas adaptatif, un octree faiblement équilibré, c'est-à-dire que deux cellules adjacentes ne diffèrent pas de plus de 1 niveau, est utilisé à la place d'une grille régulière. Les travaux de Molino et al. [Mol+03] et de Labelle et al. [LS07] illustrent la création d'une telle structure. Cette méthode a pour avantage de permettre d'avoir des tétraèdres d'une forme presque optimale, la qualité du pire élément étant bornée et en pratique totalement acceptable. Wojtan et Turk [WT08] utilisent

cette structure pour simuler des objets viscoélastiques en remaillant l'objet localement lorsque la qualité des éléments se dégrade. Sifakis et al. [Sif+07] utilisent également cette technique en introduisant des jonctions en T dans le maillage et proposent une méthode pour les gérer en les contraignant sur le bord des cellules incidentes.

Cependant, le principal problème de cette méthode est qu'elle n'autorise que l'utilisation de tétraèdres pour la simulation, et la structure en arbre limite le contrôle sur la forme de l'objet animé. Pour certaines applications, où un contrôle précis sur la forme de l'objet est nécessaire, il peut être nécessaire d'explorer d'autres approches ou extensions de cette méthode.

Adaptativité des maillages non structurés

L'adaptation peut également être mise en œuvre sur des maillages non structurés, une approche étroitement liée au problème du remaillage dans la modélisation. Ce problème a déjà fait l'objet de nombreuses études dans le contexte de la géométrie algorithmique, et les méthodes de simulation adaptative peuvent bénéficier de ces travaux en les adaptant à leurs propres besoins. En effet, l'introduction de la simulation apporte de nouvelles contraintes sur les techniques de remaillage, en particulier en ce qui concerne la qualité des éléments. Cette dernière ne doit pas être trop dégradée pour garantir une stabilité des modèles.

Au cours de l'adaptation de la discrétisation en temps réel, une vigilance particulière est requise. En effet, l'ajout de degrés de liberté peut introduire des erreurs lors du transfert de l'énergie et du momentum vers le nouveau maillage, ou créer des effets de clignotement. De même, la suppression de degrés de liberté au sein de la simulation doit être effectuée avec précaution, car le nouveau maillage grossier ne peut pas représenter précisément les déformations du maillage fin. Selon les caractéristiques du matériau simulé, différentes méthodes de remaillage existent dans la littérature.

La première catégorie de méthodes consiste à utiliser un schéma de raffinement fixe c'est le cas des méthodes se basant en 2D sur les schémas $\sqrt{3}$ ou la bissection d'arête pour la subdivision de triangle ou en 3D sur des schémas exploitant la subdivision de tétraèdre comme dans les travaux de Koschier et al. [KLB15]. Wu et al. [Wu+01] proposent également d'adapter les maillages progressifs que nous avons vus dans une partie précédente pour pré-calculer des paramètres pour les éléments finis.

Lorsque la dégradation de la qualité des éléments est indésirable ou pourrait affecter négativement la précision et la stabilité de la simulation, une alternative intéressante est l'utilisation de maillages non imbriqués à différentes résolutions comme dans les travaux de Debunne et al. [Deb+00 ; Deb+] ces schémas adoptent une approche fondamentalement différente de la stratégie de raffinement hiérarchique.

Chaque niveau de résolution est représenté par un maillage complet indépendant qui n'a pas nécessairement de nœuds en commun avec les maillages des autres niveaux. Ces maillages peuvent être optimisés a priori, c'est-à-dire avant le début de la simulation, en fonction des exigences spécifiques du problème à résoudre. Par exemple, dans des régions où l'on s'attend à des variations plus importantes ou à des détails plus fins, un maillage plus raffiné pourrait être préparé.

Pendant l'exécution de la simulation, différentes régions du domaine de simulation utilisent des sous-ensembles de différents maillages. La continuité entre ces régions est assurée en permettant une légère superposition des maillages de résolutions différentes. Les nœuds dans cette région de chevauchement sont considérés comme des nœuds *fantômes* inactifs. Ces nœuds sont incorporés dans l'élément du maillage alternatif, assurant ainsi une transition en douceur et sans heurts entre les niveaux de résolution. Une caractéristique de cette méthode est qu'elle s'intègre harmonieusement avec les approches multi-grilles et les techniques de détection de collisions adaptées à l'adaptabilité comme on peut le voir dans les travaux de Otaduy et al. [Ota+07] qui obtiennent des simulations précises, en temps interactifs sur des maillages très détaillés.

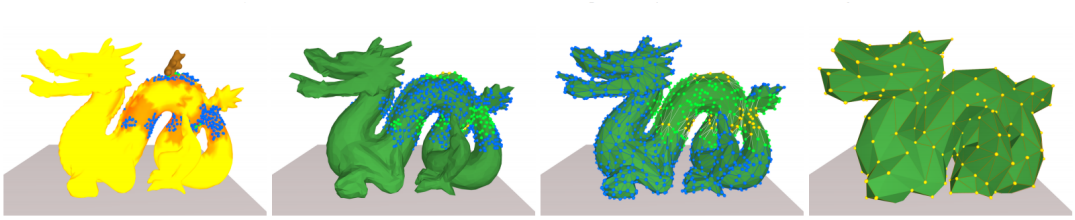

FIGURE 4.12 – Illustration de la méthode de Otaduy et al. les détails sont ajoutés proche du point de contact avec l'objet. (issu de [Ota+07])

Dans certains contextes, il est souhaitable de modifier librement la connectivité du maillage pendant le déroulement de la simulation. C'est le cas, par exemple, lorsque des éléments anisotropes sont nécessaires pour résoudre des caractéristiques fortement directionnelles, ou lorsque le matériau présente à la fois des propriétés élastiques et une déformation illimitée, comme dans un écoulement plastique. L'un des défis est alors de préserver la qualité du maillage. Les éléments du maillage doivent être bien conditionnés pour ne pas dégrader la stabilité de la simulation. De nombreuses mesures de qualité des éléments ont été proposées à cet effet. En dimension 2, la triangulation de Delaunay optimise plusieurs notions importantes de qualité du maillage, mais en dimension 3, la tétraédrisation de Delaunay offre peu de garanties de ce type, et des stratégies d'amélioration du maillage plus sophistiquées peuvent être nécessaires.

Il existe deux approches du remaillage : global ou local. Le remaillage global consiste à créer un nouveau maillage de simulation à partir de zéro chaque fois que cela est nécessaire. Cette approche est simple et directe, mais elle peut conduire à une diffusion indésirable des quantités physiques stockées, comme les informations de plasticité. Le remaillage local est une alternative de plus en plus populaire qui modifie le maillage en se concentrant sur des zones spécifiques plutôt que sur l'ensemble du maillage. Les techniques de remaillage local utilisent un ensemble d'opérations locales qui affinent, grossissent et modifient la forme des éléments existants. Cette approche est plus délicate, mais elle offre une plus grande précision et un meilleur contrôle sur le processus d'adaptation. Les travaux de Narain et al. [NSO12] montre l'utilisation de remaillages locaux en 2D et les travaux de Wicke et al. [Wic+10] l'illustre en 3D.

L'adaptation des maillages non structurés constitue un domaine en évolution rapide avec des défis uniques en simulation. Les méthodes varient selon le contexte et les matériaux simulés, et une attention particulière doit être accordée à la qualité et à la stabilité du maillage. Les techniques récentes offrent des avantages tels que la réduction de la diffusion numérique et une représentation explicite de la surface, enrichissant ainsi les possibilités dans la modélisation et l'animation.

Raffinement de base

Nous avons examiné des méthodes qui modifient la connectivité de l'objet pour introduire davantage de degrés de liberté dans la simulation. Toutefois, nous avons observé que ce remaillage peut poser des problèmes sur la qualité des éléments. De plus, modifier le maillage initial peut s'avérer indésirable selon l'application. Pour surmonter ce défi, il est possible de raffiner les fonctions de base utilisées, en particulier dans la méthode des éléments finis. Par exemple, une simulation recourant aux éléments finis utilise généralement des fonctions de base linéaires sur les éléments. Un raffinement de base peut alors consister à utiliser localement des fonctions de base quadratiques pour améliorer la qualité de la simulation sans modifier la discrétisation spatiale de l'objet.

Une première catégorie de méthodes utilise des fonctions de base hiérarchiques. C'est le cas du framework CHARMS (Conforming Hierarchical Adaptive Refinement Methods) de Grinspun et al. [GKS02]. Ce modèle gère l'apparition de jonctions en T directement au niveau des fonctions de base. CHARMS propose de nombreux exemples intéressants utilisant ces bases hiérarchiques. Cependant, il est important de noter que les différents nœuds ne peuvent pas être placés de manière arbitraire et doivent suivre un schéma précis. L'objet à simuler doit donc être plongé dans un maillage de simulation.

Ensuite, on trouve les méthodes d'enrichissement polynomial de base. Dans ce cas, l'objectif est de remplacer les fonctions de base par des fonctions de degré supérieur, par exemple remplacer des fonctions linéaires par des fonctions quadratiques. C'est le cas de Bargteil et Cohen [BC14] qui utilisent des éléments de Bézier linéaires et quadratiques. Cependant, la différence de niveau de résolution entre deux régions est limitée.

Enfin, il existe des méthodes permettant de remplacer des fonctions de base par d'autres fonctions plus pratiques dans un certain contexte. C'est le cas des XFEM [BGV09], qui permettent par exemple de représenter une fracture directement au niveau des fonctions de base.

Cette catégorie de méthodes qui utilisent un raffinement au niveau des fonctions de base n'exclut évidemment pas les méthodes utilisant un raffinement géométrique de l'objet. Notamment en présence de contacts importants, il peut être judicieux de combiner ces deux approches. Comme indiqué dans l'état de l'art de Manteaux et al. [Man+17], l'utilisation de ces deux approches est une piste très prometteuse pour de futures recherches.

4.4 Découpe et déchirure

Dans de nombreuses applications interactives, la prise en compte des modifications de la structure topologique des objets est essentielle pour la simulation de coupures, de déchirures ou de fractures. Ce sujet a suscité un champ de recherche actif, dont les travaux peuvent être catégorisés en deux approches principales : la modification directe de la topologie de l'objet animé et l'utilisation de techniques de reconstruction pour créer une surface à partir d'un maillage grossier, sur lequel les simulations et les coupures sont réalisées.

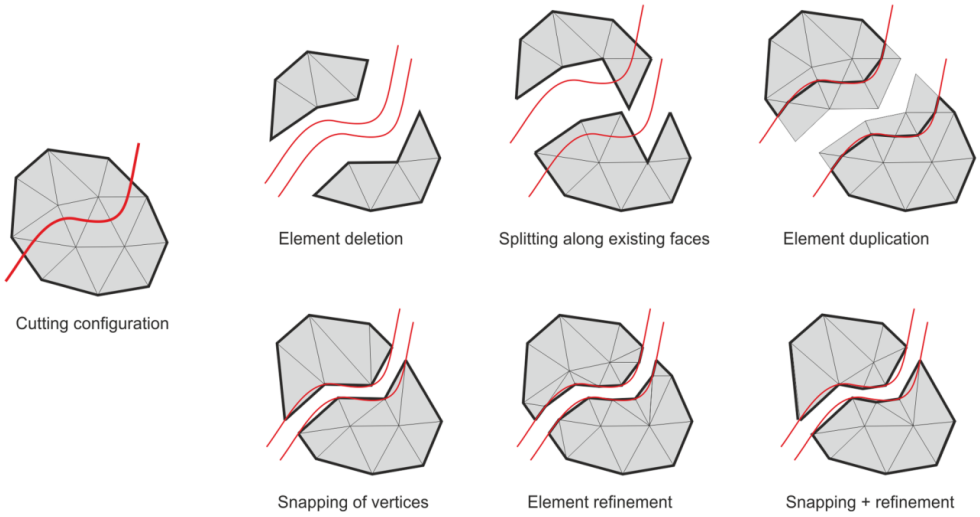

FIGURE 4.13 – Méthodes de découpe basées remaillage (Issu de [WWD15])

Les méthodes qui agissent directement sur les objets sont basées principalement sur des approches de remaillage local [WWD15]. Les opérations de remaillage varient de la suppression des éléments coupés [CDA00] à la séparation des faces et des arêtes sur la surface de coupe [MG04], ou encore à la subdivision des éléments coupés [Bie+03].

Ces techniques, lorsqu'elles sont appliquées de manière simple, engendrent des surfaces de séparation non lisses. Des améliorations ont été proposées, telles que le *snapping* (déplacement des sommets du maillage sur le plan de coupe) [NS01 ; Ste+06] ou l'utilisation de schémas de subdivision spécifiques[KLB15 ; Pau+15]. Cependant, bien que ces améliorations soient efficaces en termes de géométrie, elles ne permettent pas de contrôler la distribution des degrés de liberté (DoFs) dans le modèle et peuvent aboutir à des éléments de mauvaise qualité.

L'autre catégorie d'approches opère sur le maillage grossier, qui est le maillage sur lequel la simulation est effectuée. Ces méthodes utilisent des algorithmes de reconstruction de surface qui travaillent sur ces mailles ou leurs structures hiérarchiques sous-jacentes [DGW11].

L'avantage de ces méthodes est qu'elles utilisent une structure régulière simple pour la simulation physique, ce qui peut être très efficace. Cependant, elles nécessitent une reconstruction de la surface de rendu à chaque pas de temps, ce qui limite les possibilités pour la représentation de détails géométriques fins.

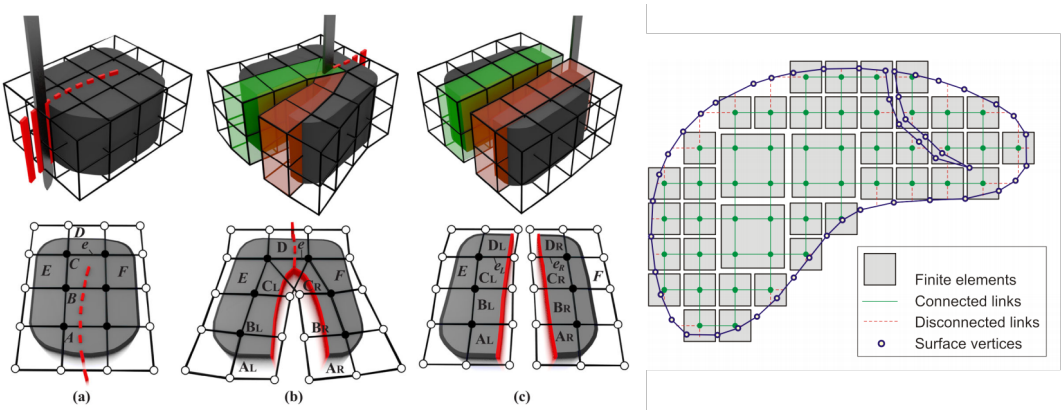

FIGURE 4.14 – Illustration des méthodes de nœuds virtuels (gauche issu de [QMD21]) et basée octree (droite issu de [WWD14]

Wu et al.[WWD14] ont proposé une discrétisation basée sur des octrees adaptatifs. La découpe est réalisée en séparant les cellules adjacentes d'un octree par l'usage de marqueurs placés sur chaque nœud. La surface de l'objet est alors reconstruite à partir de ces cellules. D'autres méthodes ont introduit le concept de *nœuds virtuels* [MBF04 ; SDF07 ; Jia+15 ; Jia+18 ; QMD21]. Ce concept consiste à dupliquer les éléments coupés en ajoutant des nœuds virtuels qui sont utilisés pour maintenir une topologie valide pour l'objet coupé. La surface est alors reconstruite à partir de ces nœuds. Ces méthodes peuvent être appliquées à des maillages structurés, comme les octrees, mais aussi à des maillages non structurés tels que les maillages tétraédriques.

Parmi les approches multi-échelles déjà mentionnées, certaines permettent la découpe ou la fracturation. Seiler et al.[Sei+11] et Steinmann et al.[SOG08] reconstruisent une structure sous-jacente lors de la découpe, tandis que Wu et al. [WWD14] utilisent une double représentation pour signaler si les cellules de l'octree sont connectées ou non, et utilisent cette représentation pour gérer les coupes dans la surface. Cependant, ces approches basées sur l'utilisation d'octrees posent plusieurs questions. Tout d'abord, elles requièrent l'établissement d'un lien entre les maillages de surface et les octrees, ce qui nécessite des mises à jour complexes et coûteuses lors de changements topologiques. De plus, l'adaptation de la structure de l'octree conduit à la formation de jonctions en T, qui peuvent poser un problème pour certaines méthodes numériques, comme les éléments finis, comme nous l'avons vu précédemment.

Chapitre 5

Contributions à l'animation d'objets détaillés

Dans le chapitre précédent, nous avons proposé un état de l'art mettant en lumière un large aperçu des problématiques liées à la simulation d'objets déformables. Nous avons observé que pour animer des objets toujours plus détaillés en temps réel, l'utilisation de méthodes adaptatives se révèle prometteuse. Ces méthodes, en ajoutant localement des degrés de liberté au sein des objets modélisés, visent à obtenir une simulation aussi précise que possible, tout en maîtrisant les temps de calcul.

Nous avons déjà proposé une structure topologique qui autorise l'activation et la désactivation des sommets d'un objet représenté par une hiérarchie multirésolution. Dans ce chapitre, nous détaillons nos contributions spécifiques à la simulation d'objets déformables, en présentant un cadre pour l'animation adaptative d'objets représentés par des vues multirésolutions.

Nous introduisons un mécanisme de propagation des déformations dans une hiérarchie multirésolution, servant de base pour proposer des algorithmes de déformation multirésolution et animer la surface fine de l'objet. De plus, nous expliquerons comment nous avons adapté différents modèles mécaniques pour les utiliser dans un cadre multirésolution, permettant l'ajout et la suppression de degrés de liberté tout en minimisant les éventuels effets de clignotement. Nous présenterons également les différents critères d'adaptation que nous avons mis en place et discuterons de leur impact sur la simulation. Enfin, nous dévoilerons un algorithme de découpe qui peut être facilement intégré dans notre cadre de travail.

5.1 Animation multirésolution adaptative

5.1.1 Propagation multirésolution des déformations

Le cœur de notre méthode d'animation adaptative réside dans l'application d'un modèle mécanique à une vue de l'objet constituée de l'ensemble de ses degrés de liberté, que nous nommerons la *vue mécanique*. Notre objectif est de pouvoir propager les défor-

mations des sommets de la vue mécanique vers des sommets plus fins dans la hiérarchie multirésolution.

Nous avons expérimenté deux méthodes de propagation des déformations dans la hiérarchie des vues. La première utilise les relations de parenté entre les sommets de la hiérarchie, tandis que la seconde s'appuie sur le gradient de déformation obtenu lors de la simulation. Nous allons maintenant détailler ces deux méthodes.

La première méthode de propagation s'appuie sur la hiérarchie multirésolution sous-jacente. Lors de la création de cette hiérarchie, chaque sommet est ajouté en subdivisant une arête, une face ou un volume. Lors de son insertion, le sommet est associé à un volume *parent*. Dans chaque volume, on stocke un repère local défini par quatre de ses sommets (non colinéaires). Les coordonnées locales de chaque sommet sont définies dans ce repère. La position d'un sommet est recalculée avec ses coordonnées et celles du volume dès que celui-ci est déformé, comme dans les travaux de Nesme et al. [NFP06]. L'algorithme de propagation des déformations s'exprime donc ainsi :

Algorithme 8 Propagation des déformations

1: Soit un sommet v.
2: **pour** chaque sommet $v2$ définissant le repère local dans lequel v est défini **faire**
3: **si** $v2$ n'a pas de position **alors**
4: Propager les déformations sur $v2$.
5: **fin si**
6: La position de v devient la position obtenue à partir du repère local.
7: **fin pour**

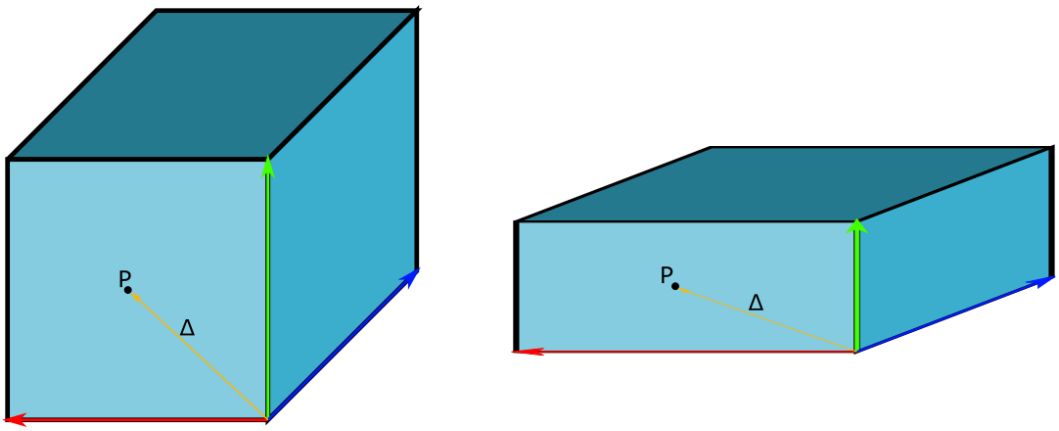

FIGURE 5.1 – Illustration de la méthode de propagation par repère local, ici le point P est défini par un vecteur Δ dans le repère de la cellule non déformé, ce même repère permet d'obtenir la position de P après déformation

Cet algorithme récursif présente l'avantage d'être très simple à mettre en place et d'être indépendant du modèle mécanique utilisé. Cependant, il a aussi des inconvénients.

Par exemple, il peut produire des déformations très importantes de la vue géométrique qui ne sont pas régies par la physique. Il nécessite également que les repères soient bien définis. En particulier, dans des configurations où les quatre points sont coplanaires ou proches de l'être, les résultats de la propagation seront peu satisfaisants. Par conséquent, le choix du critère d'adaptation, que nous aborderons plus tard, est d'une importance cruciale avec cette méthode pour éviter des cas dégénérés.

La deuxième méthode de propagation consiste à utiliser le gradient de déformation calculé par le modèle physique. En effet, chaque sommet de la vue géométrique peut être associé à un volume de la vue mécanique et il est raisonnable de considérer que le gradient de déformation est constant dans ce volume, sinon il aurait été subdivisé lors de la phase d'adaptation. Ainsi on calcule la position x_i d'un sommet associé à un volume e à partir de sa position initiale x_i^0, du centre de masse cm_e de e, du centre de masse initial cm_e^0 et du gradient de déformation F_e avec :

$$x_i = F_e * (x_i^0 - cm_e^0) + cm_e$$

Cette formule est fortement inspirée du filtrage zéro énergie utilisé dans le shape matching physique présenté précédemment. Elle a pour avantage d'utiliser le gradient de déformation déjà calculé lors de la simulation et d'apporter des résultats plus stables, surtout dans des situations géométriquement dégénérées. Enfin elle ne nécessite pas de parcourir la hiérarchie pour propager les déformations et est donc optimal dans ce sens en limitant les mises à jour géométriques aux sommets présents dans la vue géométrique.

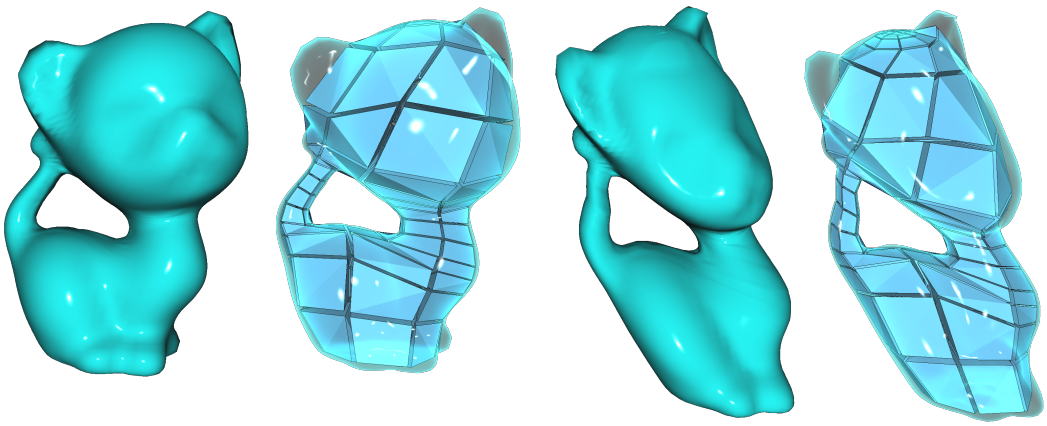

FIGURE 5.2 – Illustration de la méthode de propagation par filtrage de zéro énergie. La surface fine (gauche) est animée en fixant les degrés de liberté d'une oreille et en appliquant une force sur le reste de l'objet. Sur la vue mécanique (milieu gauche), la surface déformée (milieu droit) est obtenu par propagation des déformations de la vue mécanique déformée (droite)

5.1.2 Adaptation des modèles mécaniques

Le second composant du système d'animation gère l'adaptation de la vue mécanique, c'est-à-dire l'ajout ou le retrait de degrés de liberté à la simulation. L'ajout de degrés de liberté est un processus délicat qui implique une modification de la connectivité du maillage. Cette modification est gérée directement par notre modèle de vues multirésolution grâce à l'activation de cellules. Il reste à attribuer des quantités physiques aux nouveaux degrés de libertés et/ou aux cellules incidentes. Selon Manteaux et al. [Man+17], ce processus peut ajouter artificiellement de l'énergie au système, nuisant ainsi à la convergence de la simulation. Pour mémoire, chaque degré de liberté doit posséder une masse, une position, et une vitesse. Nous allons voir comment attribuer au mieux ces grandeurs physiques aux nouveaux degrés de liberté.

La masse est attribuée en évaluant la masse des volumes actifs, c'est-à-dire visibles dans la vue mécanique, et en la répartissant entre leurs sommets. Chaque nouveau degré de liberté hérite ainsi d'une partie de la masse de l'élément qui l'a généré. En garantissant que la somme des masses des nouveaux volumes est égale à la masse du volume avant subdivision, nous assurons une conservation globale de la masse de l'objet après raffinement. Cet équilibre des masses est garanti lors de la génération de la hiérarchie. La masse des volumes est calculée au niveau le plus fin à partir de la masse volumique et du volume de chaque élément au repos. Puis, de manière récursive, la masse de chaque volume est calculée en sommant les masses de ses sous-volumes.

La vitesse est définie comme la moyenne des vitesses des éléments auxquels les nouveaux degrés de liberté appartiennent, la vitesse d'un élément étant la moyenne des vitesses de ses sommets. Cette approche permet de conserver l'énergie cinétique de chaque élément du maillage.

L'attribution des positions est peut-être la partie la plus complexe de ce processus. Il est nécessaire de garantir une cohérence entre la position des nouveaux degrés de liberté et leur position s'ils avaient été obtenus par propagation. Cela évite des effets de clignotement sur le bord de l'objet lors du passage d'un sommet de mécanique à un sommet uniquement géométrique. Ainsi, la position des nouveaux degrés de liberté est obtenue à l'aide de l'opérateur de propagation que nous avons précédemment examiné. Cette méthode s'avère particulièrement pertinente lors de l'utilisation du filtrage zéro énergie, car l'ajout de nouveaux degrés de liberté suivant le gradient de déformation assure la stabilité du système, sans ajouter d'énergie supplémentaire. Il est important de noter qu'en utilisant cette méthode avec des modèles mécaniques non physiques tels que le shape matching classique, nous ne pouvons pas garantir la conservation de l'énergie du système.

La suppression de degrés de liberté est, quant à elle, beaucoup plus simple. L'aspect topologique est à nouveau directement géré par notre modèle de vues multirésolution. Nous pouvons partir du principe que si des degrés de liberté sont identifiés comme pouvant être supprimés, les contraintes sur cette partie du maillage sont relativement uniformes. Il est alors nécessaire de mettre à jour la masse des éléments incidents aux sommets supprimés. Cependant, la position et la vitesse de ces mêmes sommets incidents n'ont pas besoin d'être modifiées, ce qui simplifie considérablement le processus.

Le processus d'adaptation du modèle mécanique se déroule comme suit :

1. Ajouter/Supprimer des sommets dans le maillage mécanique
2. Donner une position et une vitesse aux nouveaux sommets
3. Mettre à jour la masse des nouveaux sommets et de leur voisinage topologique
4. Mettre à jour l'ensemble des plongements de cellules nécessaires au modèle mécanique utilisé

Pour utiliser un nouveau modèle mécanique, il est donc nécessaire de définir comment mettre à jour les différentes structures utiles au modèle, en fonction des degrés de liberté ajoutés ou retirés de la simulation. Nous allons maintenant illustrer cette méthode pour trois modèles mécaniques que nous avons implémentés.

Shape Matching par régions

Le premier modèle implémenté au cours de cette thèse est le modèle du *Shape Matching* par régions, présenté précédemment. Pour cette méthode, nous avons choisi de conserver des régions de taille fixe correspondant au 1-voisinage topologique de chaque sommet. L'ajout et la suppression de degrés de liberté doivent donc s'accompagner d'une mise à jour des informations de leurs régions, notamment leur masse et leur centre de masse initial.

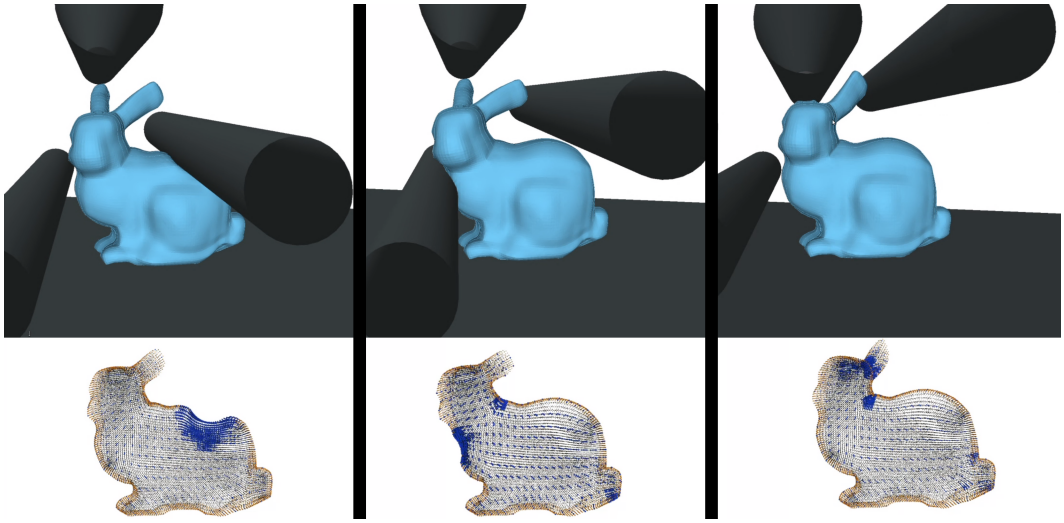

FIGURE 5.3 – Simulation adaptative utilisant le *Shape Matching* par régions avec des régions de taille 1. La ligne du haut représente trois pas de temps de la simulation, la ligne du bas représente une vue en coupe de l'intérieur de l'objet permettant de montrer la répartition des degrés de liberté en bleu dans le maillage, les sommets orange correspondent aux sommets de la vue géométrique et les sommets noirs aux sommets de la résolution la plus fine.

La Figure 5.3 présente les résultats de la simulation adaptative du *Shape Matching*. Étant donné que cette méthode d'animation n'est pas basé sur des principes physiques, la méthode de propagation par repère local a été utilisée à la fois pour adapter les sommets du bord et pour déterminer la position des nouveaux sommets.

Bien que cette méthode soit relativement simple à mettre en œuvre, le manque de contrôle sur le comportement de l'objet en raison de son aspect purement géométrique pose des problèmes en ce qui concerne la prétendue convergence vers une vérité terrain aux résolutions les plus fines. En effet, les régions de taille fixe font que les résolutions les plus fines représentent un objet de moins en moins rigide.

Shape Matching physique

Le deuxième modèle physique que nous avons mis en place est le *Shape Matching* basé physique. Il présente l'avantage d'utiliser de vrais paramètres physiques pour simuler le comportement de l'objet, ce qui permet de garantir la conservation du volume de l'objet et d'assurer une convergence de la simulation à mesure que le nombre de degrés de liberté augmente. Les résultats de la simulation adaptative utilisant le *Shape Matching* physique sont visibles dans la Figure 5.4.

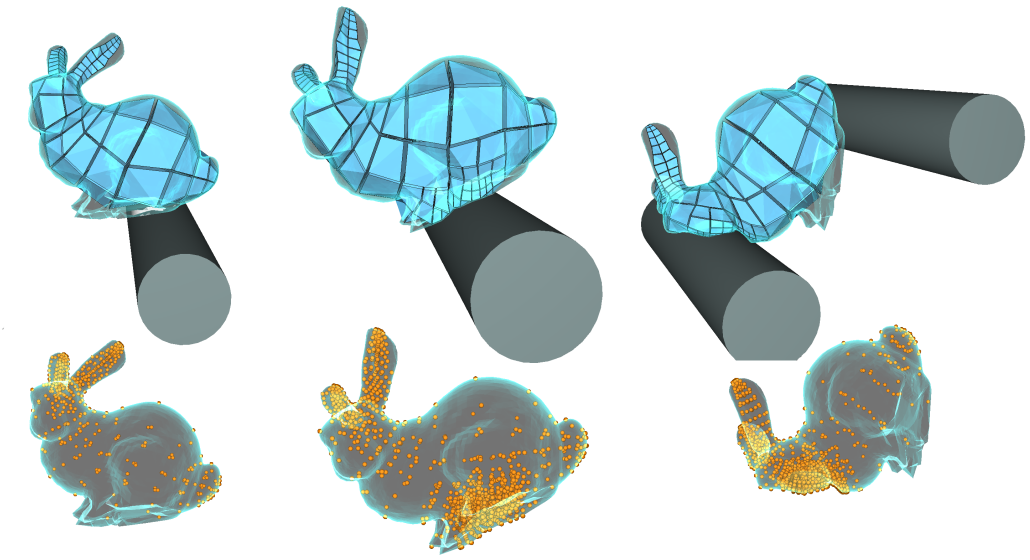

FIGURE 5.4 – Simulation adaptative utilisant le *Shape Matching* physique. La surface obtenue en utilisant le filtrage zéro énergie est visible en transparence au-dessus du maillage mécanique (en haut) et des degrés de liberté (en bas).

En ce qui concerne les mises à jour du modèle, elles sont similaires à celles du *Shape Matching* par région. Il est nécessaire de mettre à jour, à des fins d'optimisation, le centre de masse initial de chaque élément, ainsi que la matrice Q^{-1}, qui reste constante

dans le temps et est utilisée pour le calcul du gradient de déformation. Le gradient de déformation est calculé lors de la résolution du *Shape Matching* physique et est également utilisé pour déterminer la position de la surface de l'objet et des nouveaux degrés de liberté.

SPH

Le dernier modèle que nous avons mis en place est le modèle SPH (Smoothed-Particle Hydrodynamics) pour la simulation d'objets déformables. Il partage avec le *Shape Matching* physique l'avantage d'utiliser les propriétés physiques de l'objet pour la simulation, permettant ainsi une convergence de la simulation lors de l'adaptation et offrant un réalisme physique accru.

Tout comme pour le *Shape Matching* physique, les seules données que nous devons mettre à jour lors de l'adaptation sont celles utilisées pour optimiser le traitement des algorithmes. Cela inclut le voisinage topologique de chaque particule, ainsi que la matrice de correction L_i pour chaque particule. Les résultats de la simulation utilisant la méthode SPH sont fortement similaires aux résultats obtenus avec le *Shape Matching* physique. Cela nous a permis de démontrer la pertinence de notre méthode pour différents modèles mécaniques. Toutefois, il convient de noter que des modèles plus dépendants de la connectivité du maillage, tels que les éléments finis, nécessiteraient une plus grande attention, notamment pour gérer les jonctions en T qui peuvent apparaître dans la simulation.

5.2 Implantation de critères d'adaptation

Comme mentionné précédemment, les méthodes de propagation des déformations dans la hiérarchie que nous utilisons sont basées sur l'hypothèse que les parties de l'objet sur lesquelles elles sont appliquées sont faiblement déformées. L'objectif d'un critère de déformation est donc de déterminer les zones sur lesquelles cette adaptation est la plus pertinente.

Bien que divers critères aient déjà été présentés dans les sections précédentes, chacun pouvant être appliqué à notre méthode, nous avons choisi de mettre en place trois autres critères tirant parti de l'aspect multirésolution et multi-vue de notre méthode.

Pour illustrer et tester ces critères, nous avons développé trois scénarios d'animation. Le premier simule les contacts entre un objet et un instrument contrôlé par l'utilisateur. Les instruments sont modélisés par des cylindres dans la scène virtuelle. Dans ce scénario, l'adaptation de l'objet animé est basée sur l'apparition de contacts. Le second scénario propose une adaptation de l'objet en fonction du point de vue. Pour cela, une caméra virtuelle, représentée par une sphère, est mise en mouvement autour de l'objet. Les cellules du maillage sont raffinées en fonction de leur distance à la caméra. Enfin, le troisième scénario montre les interactions d'un objet avec un environnement virtuel et inclut une adaptation d'objet dépendant de ses déformations.

5.2.1 Critères basés sur l'interaction

Nos deux premiers critères sont des critères non physiques, basés sur les interactions de l'objet avec le monde extérieur. Le premier critère, le plus simple, est basé sur la proximité : lorsqu'un objet est proche de notre maillage, les éléments les plus proches de cet objet sont subdivisés. Cette méthode est notamment utile pour affiner la simulation dans les zones visibles par la caméra, comme illustré dans la Figure 5.5. En effet, dans le cadre d'animations physiquement plausibles où la qualité graphique de l'animation est plus importante que son réalisme physique, cette méthode peut être très utile. Outre l'adaptation de la vue mécanique, elle peut également être utilisée pour l'adaptation d'autres vues, comme la vue géométrique, qui pourrait exposer une surface fine uniquement sur les parties visibles de l'objet.

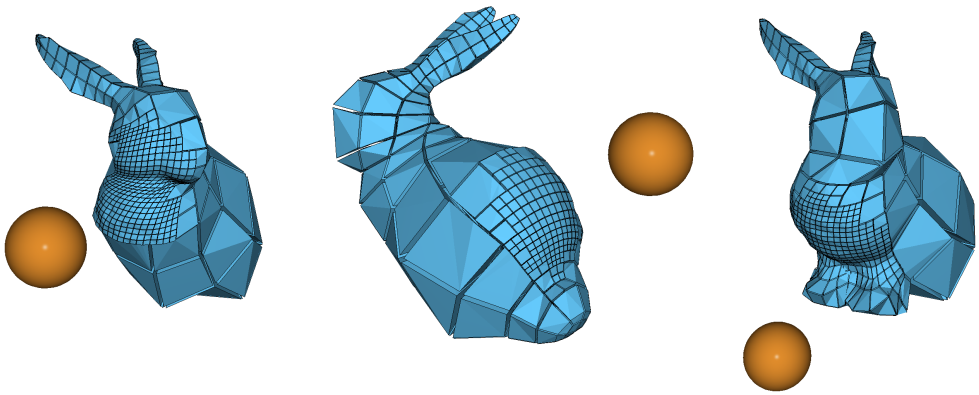

FIGURE 5.5 – Scénario de la sphère en orbite : captures d'écran de la scène montrant les adaptations induites.

Un second critère que nous avons mis en place est un critère basé sur les collisions : lorsqu'un objet entre en collision avec un autre, les éléments en contact sont subdivisés pour améliorer la précision des comportements calculés dans ces zones de contact. Lors de collision, ces zones sont celles qui comportent le plus de contraintes physiques et le plus de non-linéarités dans ces contraintes. Raffiner les cellules présentes dans ces zones répond aussi à une motivation mécanique.

Notons ici qu'une vue dédiée à la détection des contacts pourrait être mise en place avec les outils déjà mentionnés. Souvent, pour des raisons liées aux performances de l'application, la détection des contacts est effectuée en utilisant les niveaux grossiers. En fonction du schéma de subdivision utilisé et de la géométrie de la surface extérieure, les volumes des niveaux plus fins peuvent ne pas être inclus dans ceux des niveaux grossiers, ce qui peut engendrer des erreurs au niveau de la détection de collisions. Une vue adaptative répondant au critère de proximité précédent pourrait ainsi permettre de détecter plus précisément les collisions tout en évitant des calculs inutiles sur les zones les moins susceptibles de subir de collisions.

L'utilisation du critère de proximité est illustrée dans la Figure 5.6. On peut voir qu'il permet de représenter au mieux la surface de contact, bien que des effets de conservation du volume dans le reste de l'objet soient encore mal pris en compte à cause de l'aspect non physique du critère.

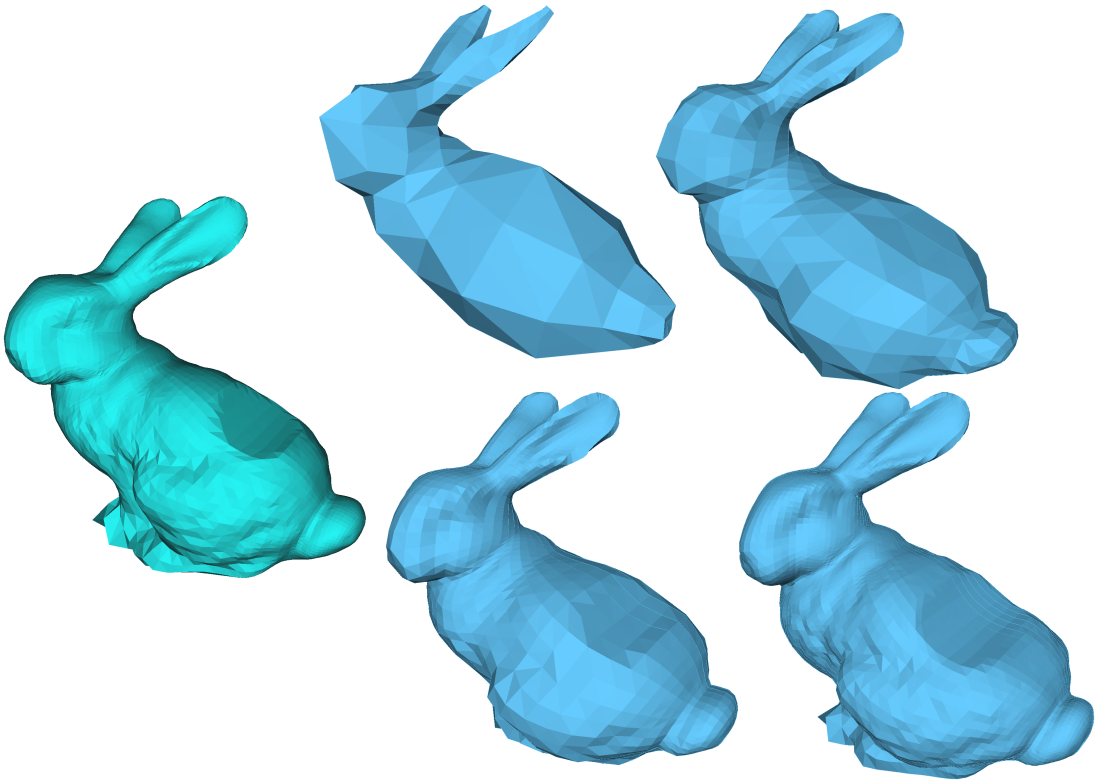

FIGURE 5.6 – Scénario de collision à l'image 50 : (à droite) déformations calculées aux niveaux 0, 1, 2 et 3 de la carte combinatoire ; (à gauche) la vue adaptative en vert montre une réponse plus précise en utilisant environ 8^3 moins de DoF.

5.2.2 Critère basé sur les aspects physiques

Nous avons vu que les méthodes basées sur les interactions permettent de concentrer les degrés de liberté de la simulation dans les zones de plus grand intérêt, mais ne permettent pas de représenter les déformations plus globales de l'objet. En particulier, dans un contexte de conservation du volume, les zones plus éloignées des points de contact ne peuvent pas se déformer correctement sans un nombre suffisant de degrés de liberté. C'est visible sur la Figure 5.6, au niveau des pattes du lapin, où la zone devrait s'élargir comme dans la version fine située à droite. À l'inverse, la tête du lapin se retrouve légèrement déformée.

Pour résoudre ce problème, nous proposons un critère d'adaptation similaire aux travaux d'Otaduy et al., où nous comparons les résultats de la simulation sur différentes versions du maillage pour déterminer les zones nécessitant des degrés de liberté supplémentaires et celles où ils peuvent être retirés avec peu d'impact sur la qualité de la simulation.

Pour ce critère, nous définissons trois vues A^-, A et A^+. A est simplement la vue mécanique telle que nous l'avons précédemment définie. A^+ est une vue constituée de l'intégralité des sommets de A, dans laquelle tous les volumes visibles ont été activés. Cela correspond en pratique à une version subdivisée une fois de plus de A. Par exemple, si A correspond au niveau de résolution 0, alors A^+ correspondra au niveau de résolution 1 et l'activation d'un élément dans A entraînera l'activation de tous ses sous-éléments dans A^+. La vue A^- est définie comme l'ensemble des volumes de A dans lequel toutes les simplifications possibles ont été réalisées. Autrement dit, si un volume est présent dans A^-, deux cas sont possibles : soit il est présent dans A, ce qui signifie qu'au moins un de ses frères dans la hiérarchie a été subdivisé, soit l'ensemble de ses fils est présent dans A. A^- correspond donc à une version plus grossière de A pour les volumes visibles de niveau supérieur ou égal à 1, les volumes de niveau 0 n'étant pas simplifiables.

Pour déterminer où adapter les degrés de liberté de la simulation, nous effectuons la simulation sur ces trois vues. Puis nous comparons la position des sommets de A après propagation vers A^+ avec la position des sommets de A^+ après application du modèle mécanique. Il est alors possible de déterminer les zones dans lesquelles l'ajout de degrés de liberté aura effectivement le plus d'impact sur la qualité de la simulation. Soit x^+ la position d'un sommet après application du modèle mécanique sur A^+ et soit Px la position du même sommet après propagation depuis A. L'erreur est obtenue comme :

$$e = ||x^+ - Px||$$

En sommant pour chaque volume l'erreur de chaque sommet, nous pouvons obtenir les volumes dont le raffinement aura le plus d'impact positif sur la qualité de la simulation. En effectuant le même travail entre A^- et A, nous pouvons classer les volumes de A pour savoir quels volumes pourront être simplifiés avec le moins d'impact sur le résultat de la simulation. L'utilisation de ces deux listes de priorité nous permet également de contrôler facilement le nombre total de degrés de liberté ainsi que le nombre de modifications topologiques effectuées simultanément, afin d'avoir un contrôle fin sur le temps de calcul de la simulation. Il est important de noter que cette méthode repose sur l'hypothèse forte que plus la résolution utilisée est localement fine, plus la simulation se rapprochera de la vérité terrain, ce qui dépend fortement du modèle mécanique utilisé.

Cette méthode a l'avantage d'être basée sur le comportement physique de l'objet et de prendre en compte l'impact effectif de l'adaptation sur la qualité de la simulation. Cependant, elle est coûteuse en termes de temps de calcul par rapport aux critères plus géométriques présentés précédemment.

5.2.3 Résultats

Afin de démontrer l'efficacité de notre méthode, nous avons mis en place trois scénarios de simulation. Dans le premier scénario, nous avons placé une sphère en mouvement autour d'un objet déformable pour expérimenter l'adaptation suivant un point de vue. Les volumes sont affinés ou simplifiés en fonction de leur distance par rapport à la sphère, le nombre total de DoF restant inférieur à la limite fixée. La sphère tourne rapidement autour de l'objet, entraînant une adaptation très rapide et significative du maillage. Au cours de cette expérience, nous avons également fait varier la limite maximale sur le nombre de DoF pour démontrer la capacité de notre modèle à contrôler les ressources de calcul utilisées.

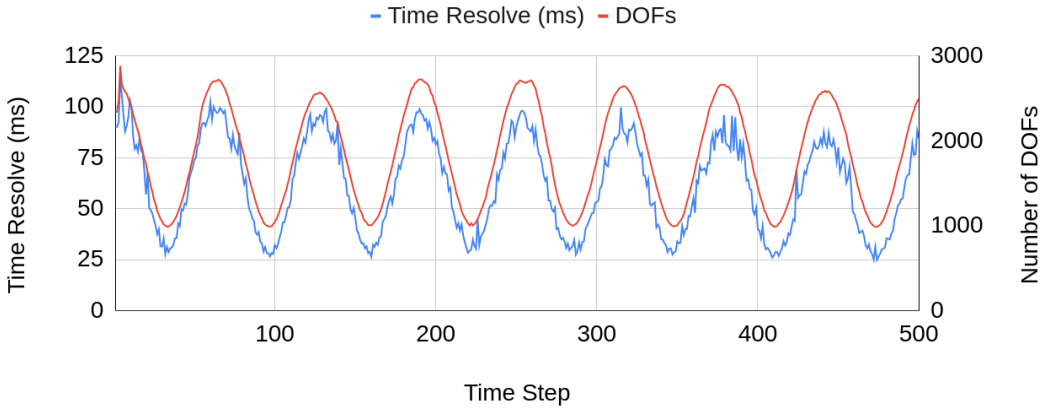

FIGURE 5.7 – Scénario de la sphère en orbite : la variation de la limite du nombre de DoF permet de contrôler les temps de calcul lors de la simulation. Dans ce cas, le nombre maximum de DoF varie de 2 à 6 fois le nombre initial de DoF.

La Figure 5.7 montre l'évolution du temps de calcul en fonction du temps, et la Figure 5.5 illustre la sphère en mouvement autour de l'objet. On peut observer sur ce graphique que notre modèle parvient à maîtriser le temps de calcul de la simulation, même lorsque le nombre et la répartition spatiale des DoF varient considérablement.

Dans le second scénario, nous avons simulé des collisions entre un objet déformable et un ensemble de cylindres contrôlés par la simulation. À chaque collision, les volumes incidents aux surfaces de contact sont affinés et les DoF sont activés en leur sein. Notre algorithme de réponse à la collision peut activer des sommets de niveau 1, 2 ou 3, offrant une réponse plus précise mais augmentant le nombre de DoF activés. La profondeur de l'affinement dépend de la limite fixée sur le nombre de DoF et du nombre et de la taille des surfaces de contact.

Pour évaluer les résultats, nous avons exécuté la simulation sur quatre maillages monorésolution à différents niveaux de résolution et sur une vue adaptative. Nous avons comparé les temps d'exécution de ces simulations et la précision des réponses. Les temps de calcul sont rapportés à la Figure 5.8, où des collisions ont lieu aux alentours des images

50 et 150. Les courbes pour les simulations effectuées aux niveaux 0, 1 et 2 montrent que les temps de calcul pour les modèles monorésolution sont constants, mais augmentent exponentiellement en fonction du niveau de résolution, un maillage de niveau i contenant en moyenne 8^i fois plus de volumes que le maillage de niveau 0.

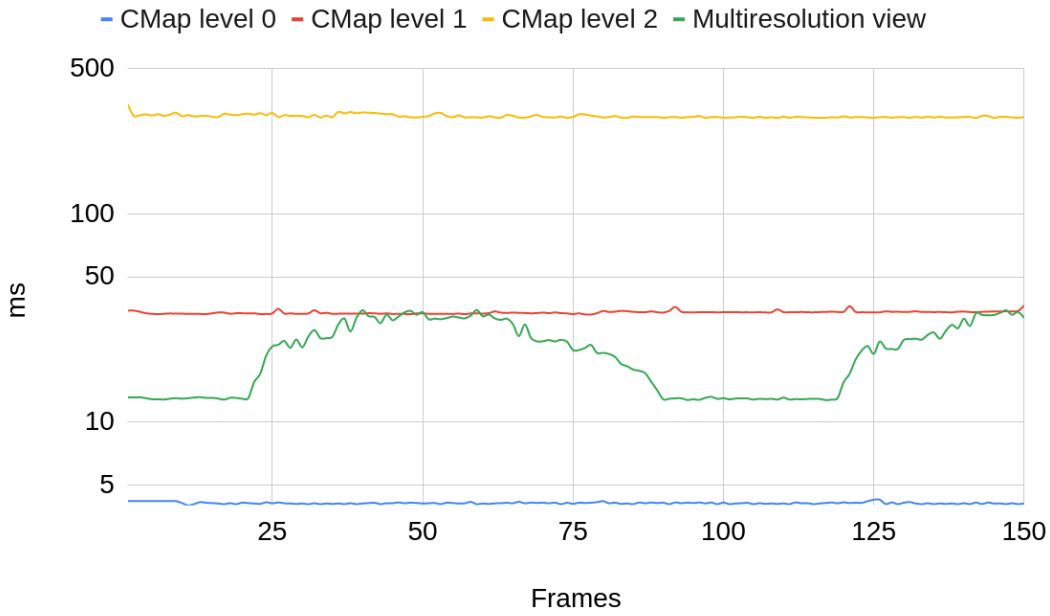

FIGURE 5.8 – Scénario de collision : évolution des temps de calcul lors d'une simulation où des collisions se produisent, pour quatre niveaux de résolution d'une carte combinatoire et pour une vue multi-résolution adaptative. L'axe vertical représente les temps de calcul en ms sur une échelle logarithmique.

La courbe verte représente les temps de calcul pour notre modèle adaptatif. Les temps de calcul augmentent pendant la collision, pour revenir à un plateau entre les performances des niveaux 0 et 1. Les temps de calcul ne dépassent jamais ceux du niveau 1, bien que dans certaines zones, les volumes soient affinés trois fois pour obtenir une réponse précise.

La Figure 5.6 illustre les déformations à l'image 50, moment où le cylindre est complètement encastré dans le lapin. Il est clair que la qualité de la réponse dépend du niveau de résolution du maillage pour le maillage monorésolution. Le résultat pour la vue adaptative est visuellement aussi bon que pour un maillage de niveau 3, avec des temps de calcul inférieurs à ceux du niveau 1. Cela démontre l'efficacité et la précision fournies par les vues adaptatives.

Dans le troisième et dernier scénario, nous avons à nouveau provoqué des collisions sur notre maillage à l'aide de cylindres. Cette fois-ci, l'adaptation est gérée grâce à notre critère d'adaptation qui gère trois vues simultanément. Ce scénario nous a permis

de démontrer les capacités de ce critère d'adaptation à détecter efficacement les zones nécessitant le plus de DoF dans la simulation.

Ces différents scénarios nous ont permis de démontrer la versatilité de notre méthode et son efficacité. À l'avenir, de nouveaux critères d'adaptation pourront être développés pour permettre une adaptation plus précise et moins coûteuse en temps de calcul.

5.3 Algorithme de découpe multirésolution

Le deuxième avantage significatif de notre structure de vues multirésolutions réside dans sa capacité à faciliter l'implantation des algorithmes de découpe, tout en préservant des propriétés précieuses pour l'animation adaptative d'objets détaillés. Ces propriétés comprennent la possibilité de faire des requêtes en temps réel sur la connectivité du maillage ou d'utiliser simultanément de multiples vues multirésolution.

Dans cette section, nous détaillons un algorithme de découpe multirésolution, en démontrant comment il s'intègre harmonieusement dans notre cadre d'animation. Nous mettons en avant les défis principaux présentés par l'implantation de cette approche et les solutions que nous avons développées pour y répondre de manière efficace.

5.3.1 Principe

Nous avons conçu un algorithme de découpe multirésolution s'appuyant sur l'approche proposée dans [DGW11]. Initialement, nous décrirons l'algorithme de découpe dans un cadre non multirésolution. Pour rappel, notre objet est défini comme un ensemble de volumes liés entre eux par leurs faces communes.

L'algorithme consiste à définir un ensemble de liens reliant les centroïdes de deux volumes adjacents par une face. Si l'un de ces liens est coupé par la surface ou l'outil de coupe, les volumes sont séparés le long de leur face commune, c'est-à-dire que les liens par ϕ_3^i entre les brins de cette face sont supprimés et de nouveaux brins, correspondant au bord de l'objet, sont créés.

Nous avons adapté cette méthode de découpe à notre modèle de vues topologiques multirésolution. Les vues nous permettent d'améliorer grandement les performances de l'algorithme de découpe. Nous choisissons un niveau i de découpe qui peut correspondre au maillage le plus fin de la hiérarchie multirésolution ou au niveau de détail désiré pour la surface de coupe. En partant d'une vue de niveau 0, il s'agit d'utiliser un parcours en profondeur dans la hiérarchie pour déterminer efficacement les volumes du niveau i qui doivent être séparés. Pour cela, nous appliquons l'algorithme suivant :

Nous commençons par appliquer notre méthode de découpe sur les volumes de niveaux 0, sélectionnant ainsi une première liste de volumes à séparer. Dans un deuxième temps, la recherche se poursuit au niveau 1, en la limitant aux enfants des volumes sélectionnés. La recherche se poursuit jusqu'à ce que le niveau de coupe souhaité soit atteint, en optimisant la détection des intersections avec la surface de coupe. Le principe de l'algorithme est présenté en 2D dans la figure 5.9.

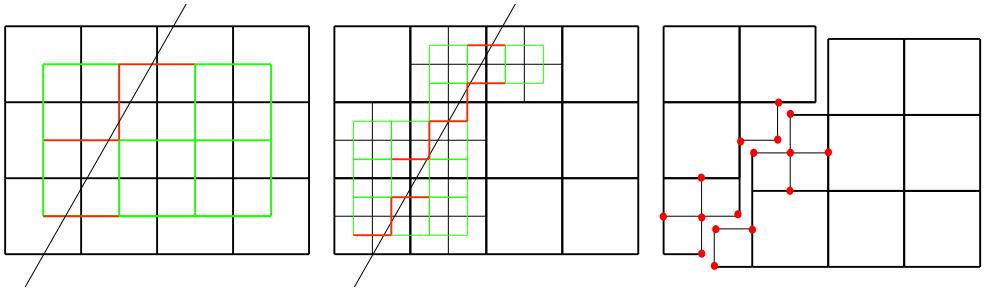

FIGURE 5.9 – Algorithme de découpe : les liens entre les paires de volumes séparés par le plan de découpe sont dessinés en rouge, les liens testés sont en vert. À gauche : la recherche au niveau 0 ; au centre : la recherche parmi les enfants au niveau 1 ; à droite : le maillage après la séparation.

Il convient également de noter que la séparation des volumes est effectuée selon les principes décrits dans la section 3.6. Ainsi, la séparation des volumes est réalisée au niveau de résolution i choisi pour la découpe, mais aussi à tous les niveaux plus fins de la hiérarchie multirésolution. Inversement, les séparations effectuées au niveau i peuvent conduire à des coupes partielles dans les niveaux plus grossiers.

5.3.2 Intégration dans le framework

Après la séparation des volumes, de nouveaux sommets, résultant de la division des sommets internes, apparaissent sur la surface. Ces sommets peuvent avoir des niveaux de résolution très différents, ce qui n'est généralement pas un problème. Cependant, dans le contexte d'une simulation interactive, il est souvent souhaitable d'intégrer ces sommets et les volumes divisés associés dans la simulation. Pour cela, nous devons rendre ces volumes et leurs sommets visibles dans la vue utilisée pour la simulation, ce qui est accompli en appliquant notre algorithme directement sur la vue topologique.

Nous pouvons ensuite associer une position au nouveau sommet en utilisant les algorithmes de propagation proposés précédemment. En revanche, cette méthode a pour principal inconvénient de suivre les faces définies dans la hiérarchie, et de ne pas exposer une surface de découpe lisse. Des travaux supplémentaires pourraient être réalisés pour déplacer les nouveaux sommets sur cette surface tout en minimisant l'apport en énergie résultant de ce déplacement.

5.3.3 Conclusion et perspectives

Cet algorithme de découpe multirésolution, intégré dans notre structure de vues topologique multirésolution, offre une facilité d'implantation tout en préservant les propriétés cruciales pour l'animation adaptative d'objets détaillés. Bien qu'il offre de nombreux avantages, il présente des limites, notamment le fait qu'il ne génère pas une surface de découpe lisse. Des améliorations futures pourraient consister à améliorer ce point, ce

qui pourrait conduire à un rendu plus précis et réaliste. Ces améliorations pourraient également inclure l'optimisation de la gestion des différents niveaux de résolution des sommets et des volumes, ce qui pourrait conduire à des performances accrues dans le contexte d'une simulation interactive. Les perspectives sont donc nombreuses et prometteuses, et nous sommes convaincus que notre approche a le potentiel pour s'améliorer encore davantage.

5.4 Conclusion

Pour résumer, l'objectif de ce chapitre était de développer un cadre multirésolution complet capable de gérer tous les aspects de la simulation adaptative d'objets déformables en temps réel. La principale contribution de ce travail est le nouveau concept de *vue adaptative*, une stratégie permettant d'adapter localement un maillage volumique au sein d'une hiérarchie multirésolution tout en conservant une cohérence topologique entre différents niveaux de résolution. Cet outil offre la flexibilité de régler dynamiquement la distribution des DoF et peut-être activé ou désactivé en fonction de divers critères et niveaux de détail. De plus, la possibilité d'utiliser des vues distinctes et indépendantes de manière concurrente offre une multitude d'applications.

Nos expériences mettent en évidence l'efficacité et l'applicabilité des vues adaptatives dans plusieurs scénarios, s'avérant particulièrement bénéfiques pour optimiser certains processus tels que la propagation des déformations jusqu'aux détails géométriques les plus fins et pour la gestion des séparations ou coupures topologiques. Notamment, bien que nos recherches se soient centrées sur les maillages hexaédriques, la polyvalence de notre cadre invite à son application à d'autres types de maillages, une perspective pour les futurs travaux.

Le modèle d'animation choisi pour notre étude était la méthode de Shape Matching basée sur la physique. Ce modèle flexible nous a permis de nous concentrer sur la résolution des défis topologiques. Bien que notre cadre soit compatible avec de nombreux modèles mécaniques, ceux qui emploient des méthodes comme les méthodes d'éléments finis nécessitent une gestion soigneuse des jonctions en T inhérentes à l'adaptation du maillage. Travailler sur la gestion des contraintes topologiques, telles que la conformité du maillage, lors de l'adaptation du modèle est donc une direction intéressante pour les travaux futurs.

Enfin, notre approche pour gérer les coupures, qui repose sur la séparation des volumes le long des faces existantes, est complémentaire à des méthodes comme le *snapping* de sommets, fréquemment utilisé pour adapter la géométrie à la surface de coupe. Les travaux futurs pourraient envisager des stratégies plus avancées, telles que le remaillage local. Cependant, cela nécessiterait le développement d'opérations de remaillage hiérarchiques spécifiques capables de maintenir la cohérence multirésolution aux côtés d'autres contraintes topologiques.

Chapitre 6

Conclusion

L'objectif de cette thèse était de proposer un cadre multirésolution complet pour gérer tous les aspects de la simulation en temps réel des objets déformables. Cela englobe l'adaptation de la distribution des degrés de liberté au sein du modèle physique, les mises à jour géométriques requises pour un rendu détaillé, ainsi que la gestion efficace des coupes ou des collisions.

Modèles topologiques adaptatifs pour l'animation

La principale contribution de cette thèse est le concept de *vue adaptative* permettant d'adapter localement la manière dont des maillages sont observés ou parcourus au sein d'une hiérarchie multirésolution. Cette adaptation s'effectue en modifiant localement la visibilité de sous-ensembles de brins. Cela permet aux applications utilisant les vues de parcourir, et donc de prendre en compte de manière transparente, des cellules rendues visibles dans des niveaux grossiers alors qu'elles sont présentes normalement dans des niveaux plus fins. L'adaptation est mise en œuvre via des opérations d'activation et de désactivation de cellules offrant une interface simple aux applications exploitant les vues.

La topologie des vues adaptatives est reconstruite à la volée à partir des informations de visibilité et de la hiérarchie sous-jacente. Cette topologie peut-être parcourue à différents niveaux de résolution en bénéficiant d'une cohérence d'ensemble entre chaque résolution. Un mécanisme d'héritage simple et efficace permet de combiner plusieurs vues pour construire des hiérarchies de vues élaborées.

Ce qui distingue le concept de vue des autres approches multirésolutions adaptatives, c'est son ancrage dans le formalisme des cartes combinatoires. Il permet la modélisation formelle et élégante de hiérarchies adaptatives de quasi-variétés. Contrairement aux structures multirésolution classiques que l'on trouve dans la littérature, notre approche permet de représenter toutes les cellules d'un objet ainsi que l'ensemble de leurs relations d'incidence et d'adjacences. Les structures classiques omettent souvent certaines cellules, c'est le cas notamment des structures à base de quadtrees ou d'octrees. Cela peut poser des problèmes lors de parcours adaptatifs en provoquant l'apparition de jonctions en T mal gérées, entraînant des problèmes de conformité des maillages manipulés.

Les cartes combinatoires multirésolution offraient déjà un cadre formel et précis pour la représentation de hiérarchies de maillages, mais ne proposaient pas de parcours adaptatifs de ces maillages. Les vues adaptatives apportent une solution souple et innovante à cette question. Elles permettent aux applications les utilisant d'agir de manière simple et efficace sur la répartition des sommets et cellules visibles à un niveau de résolution donné, tout en garantissant la cohérence de l'ensemble de la hiérarchie. Cette possibilité est particulièrement adaptée aux besoins des applications d'animation interactives d'objets déformables visées.

Dans un second temps, nous avons démontré l'efficacité des vues, de leur implantation et des algorithmes de reconstructions proposés. Pour cela nous avons proposé une série de tests consistant à appliquer des traitements géométriques, nécessitant un grand nombre de requêtes de voisinage, sur des vues multirésolution et des cartes classiques, tout en activant et désactivant une partie des cellules des maillages manipulés. Les résultats de ces tests montrent que le surcoût induit par la reconstruction à la volée des relations topologiques est d'autant plus faible que le nombre de niveaux de résolution, et donc la taille des objets manipulés, augmente. Cela nous a rassuré sur la pertinence de leur mise en œuvre au sein d'applications interactives. Le résultat le plus important est lié au phénomène de fragmentation de la mémoire que nous avons mis en évidence. Celui-ci a un impact significatif sur la performance des applications visées. La gestion "en place" de la visibilité au sein des vues adaptatives apporte une réponse particulièrement efficace à ce problème. La stabilité dans le temps des performances des vues adaptatives surclasse celle des modèles monorésolutions classiques qui montrent ici leur limite.

Animation interactive d'objets détaillés

Dans la seconde partie de cette thèse, nous avons expérimenté l'utilisation de vues adaptatives dans le cadre de l'animation interactive d'objets déformables très détaillés. Pour démontrer la généricité de notre approche, nous avons implanté et adapté trois méthodes d'animation dont l'usage est répandu en informatique graphique. Nous avons également testé et implanté plusieurs critères et méthodes d'adaptation, ainsi qu'un mécanisme de gestion des découpes et déchirures au sein des objets manipulés.

Les méthodes d'animation implantées sont le *Shape Matching* qui estime les forces internes au sein d'un objet en comparant les positions courantes des sommets à leurs positions au repos ; le *Physically based Shape Matching* qui étend le Shape Matching en reformulant le calcul du gradient de déformation via un modèle néo-Hookéen représentant plus fidèlement le comportement mécanique des objets ; et enfin une variante des *Smoothed Particle Hydrodynamics* (SPH) permettant la simulation d'objets déformables.

Le principe général du moteur d'animation consiste à définir une *vue mécanique* constituée de l'ensemble de sommets actifs qui servent de degrés de liberté au modèle physique. Cet ensemble, initialement constitué de tous les sommets de niveau 0, évolue au cours de la simulation en activant ou désactivant des cellules, selon les critères définis. Dans tous nos scénarios, nous avons fixé une limite supérieure au nombre de sommets actifs afin de contrôler le niveau de performance.

Les déformations sont calculées sur les sommets actifs, puis propagés des niveaux de résolution les plus grossiers, jusqu'au niveau le plus fin qui encode les détails géométriques. Nous avons développé deux méthodes de propagation des déformations dans la hiérarchie. La première méthode exploite des repères locaux définis dans chaque cellule et la relation de parenté existant entre les sommets de la hiérarchie. La deuxième utilise le principe du filtrage zéro énergie déjà utilisé dans le Shape Matching basé physique. Pour cette méthode, nous avons montré comment exploiter une *vue géométrique* constituée des sommets du bord du niveau le plus fin, pour propager directement, et donc plus efficacement, les déformations calculées pour la vue mécanique.

Nous avons ensuite développé et testé différents critères d'adaptation permettant de déterminer où et quand ajouter ou retirer des degrés de liberté dans la simulation. Ces critères se classent en plusieurs catégories : les critères géométriques basés sur le niveau de déformation des différents types de cellules (arêtes, faces ou volumes) ; les critères physiques exploitant le gradient de déformation des cellules ; les critères liés aux contacts ou collisions détectés au sein de l'environnement virtuel ; et enfin les critères liés au point de vue plaçant les efforts de calcul sur les parties visibles de l'objet.

Nous avons enfin expérimenté notre moteur d'animation dans trois scénarios d'animation exploitant les modèles et critères d'adaptation implantés. Le premier scénario simule des contacts entre un objet déformable et un ensemble de cylindres pilotés par l'animation. L'adaptation consiste à raffiner le maillage mécanique dans les zones de contact. Ce scénario nous a permis de valider la capacité de notre méthode à améliorer la précision des calculs au niveau des surfaces de contact. Les résultats sont visuellement similaires à ceux obtenus avec une simulation réalisée au niveau de résolution 3, avec des meilleures performances que celles observées au niveau 1.

Le deuxième scénario simule le déplacement d'une caméra autour d'un objet. L'adaptation consiste à raffiner le maillage en fonction de la distance à cette caméra. Dans ce scénario, nous avons également fait varier le nombre maximal de sommets actifs autorisés dans la simulation. Nous avons ainsi pu constater que nous étions en mesure de contrôler finement les temps de calcul alloués à la simulation, et que ce temps de calcul était étroitement corrélé avec le nombre de sommets actifs.

Dans le troisième scénario, nous exploitons deux vues mécaniques parcourant les sommets actifs à deux niveaux de résolution différents. La simulation est exécutée simultanément sur les deux vues et les différences observées entre elles sont utilisées comme une mesure d'erreur utilisée comme critère d'adaptation du maillage. Ce scénario a démontré la capacité des vues à permettre des parcours adaptatifs, concurrents et réalisés à différents niveaux de résolution d'un même objet.

Enfin, nous avons développé un scénario simulant des découpes et déchirures au sein d'un objet exploitant une *vue topologique* permettant de rendre visibles dans toutes les autres vues les sommets présents sur la surface de coupe quelque soit le niveau de résolution dans lequel ces coupes ont été réalisées. Nous avons ainsi démontré l'intérêt de pouvoir combiner les vues au sein d'une hiérarchie de vues et la pertinence de notre approche pour gérer efficacement et élégamment des changements topologiques intervenant dans les objets manipulés.

Perspectives

L'ensemble des travaux réalisés au cours de cette thèse ouvre la voie à de nombreuses perspectives. Elles se répartissent suivant différentes thématiques correspondant à l'amélioration ou l'enrichissement, à court ou moyen terme, des contributions recensées dans les paragraphes précédents, mais aussi à des prolongements à plus long terme des travaux présentés.

Nature des maillages et hiérarchies considérés

Bien que les vues adaptatives et leurs propriétés aient été définis de manière très générale, les opérations d'activation et de désactivation de cellules et les algorithmes de reconstructions des ψ_i^k n'ont été présentés que dans le cadre de l'utilisation de maillages hexaédriques et d'une hiérarchie basée sur un schéma de subdivision classique. Le cadre de travail présenté dans cette thèse n'en reste pas moins générique et polyvalent.

Il serait intéressant d'étendre les opérations et algorithmes proposés à d'autres types de maillages et de hiérarchies multirésolutions. Nous pensons notamment aux maillages tétraédriques, aux hiérarchies construites par bissection d'arête, fusion de sommets ou fusion de tétraèdres, et aux schémas de subdivision associés à ce type de maillage, comme le schéma $\sqrt{3}$. Pour chaque type de hiérarchie, il faudra définir des contraintes topologiques sur les vues, adapter les opérations d'activation et de désactivation de cellules et proposer des algorithmes de reconstruction des relations topologiques spécifiques.

Les maillages tétraédriques sont largement utilisés dans le domaine de l'animation temps réel. Proposer un cadre adaptatif, générique et efficace pour l'ensemble des modèles mécaniques basés sur des tétraèdres ouvre des perspectives intéressantes.

Gestion des découpes

L'amélioration de la gestion des découpes ou déchirures au sein des maillages utilisés amène le même type de perspectives. Dans cette thèse, nous avons utilisé une méthode de découpe relativement simple consistant à séparer le volume le long de faces existantes. Cette approche est compatible avec la technique dite du *vertex snapping*, couramment utilisée pour adapter la géométrie à la surface de coupe.

Il existe cependant des méthodes plus sophistiquées, comme celles basées sur un remaillage local, dont nous pourrions envisager l'implantation. Cela nécessiterait le développement d'opérations de remaillage hiérarchique, compatibles avec les contraintes topologiques imposées par l'utilisation des vues adaptatives, et respectant la hiérarchie multirésolution en place. Ces perspectives enrichissent celles déjà mentionnées, en élargissant la nature des opérations applicables à nos structures multirésolutions.

Nature des modèles physiques utilisés

Nous avons choisi d'implanter des modèles d'animation compatibles avec l'approche adaptative proposée. Il s'agit pour l'essentiel de méthodes dites *meshless* qui supportent

naturellement les polyèdres générés par l'activation adaptative de cellule. Notre approche est cependant compatible avec d'autres méthodes de simulation qui auraient cependant demandé des développements plus lourds pour gérer correctement les jonctions en T qui apparaissent dans les méthodes adaptatives. Nous pensons notamment à la méthode des éléments finis dont l'implantation exigerait une gestion plus subtile de l'activation de cellules en vue de minimiser l'apparition de jonctions en T dans le maillage. Les méthodes à base de FEM peuvent prendre en compte les jonctions en T, mais cela nécessite d'utiliser des méthodes numériques coûteuses pour prendre en compte ces contraintes géométriques.

Pour limiter les surcoûts liés aux nombres et à la répartition des jonctions en T, plusieurs pistes sont envisageables. Il serait par exemple possible limiter le nombre de niveaux de résolution d'écart entre deux volumes adjacents, pour éviter l'apparition de plus d'un sommet au milieu d'une face. Une gestion des activations de cellules par blocs compacts pourrait également réduire la dissémination des jonctions en T, en limitant leur présence à l'interface entre deux blocs de résolution différente. Il serait enfin intéressant d'utiliser des opérateurs de subdivision plus *sobres*, comme la division d'un hexaèdre en deux, qui réduiraient le nombre de jonctions en T générées. De manière plus générale, la gestion de contraintes topologiques, spécifiques aux méthodes numériques utilisées, représente une piste également prometteuse pour de futures recherches.

Implantation GPU des vues adaptatives

De nombreux travaux en informatique graphique portent sur le portage sur GPU d'algorithmes initialement développés de manière classique sur CPU. Plusieurs méthodes d'animation ont déjà été portées sur GPU, mais, pour le moment, leur implantation se limite à l'animation de maillages dont la topologique reste constante, ce qui interdit les adaptations dynamiques et la gestion de découpes.

Des travaux récents [DV21 ; VD22] ont proposé une implantation GPU de la structure topologique des demi-arêtes pour générer des surfaces de subdivisions, via le schéma de Catmull-Clark, sur cartes graphiques. La structure hiérarchique utilisée, basées sur le modèle des demi-arêtes, est semblable aux cartes combinatoires multirésolutions de dimension 2. Ces travaux rendent donc envisageable le portage sur GPU des 3-cartes multirésolution exploitant un schéma de subdivision équivalent. Il serait alors possible de proposer une implantation des vues adaptatives qui ont, comme nous l'avons vu, la propriété de modifier la topologie "en place", c'est-à-dire sans nécessiter de modifier la position en mémoire des informations de connectivité. Il serait alors possible de proposer la gestion de maillages dynamiques et de changements topologiques sur GPU.

L'approche et les modèles combinatoires utilisés dans cette thèse ne suffiront pas, à eux seuls, à résoudre les problèmes soulevés par les différentes perspectives mentionnées. Nous sommes convaincus cependant qu'ils faciliteront le travail en fournissant un cadre de travail formel et robuste, propice aux développements de solutions innovantes.

Bibliographie

[All+07] Jérémie Allard et al. « Sofa-an open source framework for medical simulation ». In : *MMVR 15-Medicine meets virtual reality*. T. 125. tex.organization : IOP Press. 2007, p. 13-18 (cf. p. 18).

[BC14] Adam W Bargteil et Elaine Cohen. « Animation of deformable bodies with quadratic Bézier finite elements ». In : *ACM Transactions on Graphics (TOG)* 33.3 (2014). Publisher : ACM New York, NY, USA, p. 1-10 (cf. p. 95).

[Bau75] Bruce G Baumgart. « A polyhedron representation for computer vision ». In : *Proceedings of the May 19-22, 1975, national computer conference and exposition*. 1975, p. 589-596 (cf. p. 19).

[BGV09] Ted Belytschko, Robert Gracie et Giulio Ventura. « A review of extended/generalized finite element methods for material modeling ». In : *Modelling and Simulation in Materials Science and Engineering* 17.4 (2009). Publisher : IOP Publishing, p. 043001 (cf. p. 95).

[Bie+03] D Bielser et al. « A state machine for real-time cutting of tetrahedral meshes ». In : *11th Pacific Conference on Computer Graphics and Applications, 2003. Proceedings.* IEEE, 2003, p. 377-386 (cf. p. 96).

[BL99] Javier Bonet et T-SL Lok. « Variational and momentum preservation aspects of smooth particle hydrodynamic formulations ». In : *Computer Methods in applied mechanics and engineering* 180.1-2 (1999). Publisher : Elsevier, p. 97-115 (cf. p. 87).

[BMF05] Robert Bridson, Sebastian Marino et Ronald Fedkiw. « Simulation of clothing with folds and wrinkles ». In : *ACM SIGGRAPH 2005 courses*. 2005, 3-es (cf. p. 89).

[CDA00] Stéphane Cotin, Hervé Delingette et Nicholas Ayache. « A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation ». In : *Visual Computer* 16.8 (2000), p. 437-452 (cf. p. 96).

[Deb+99] Gilles Debunne et al. « Interactive multiresolution animation of deformable models ». In : *Computer animation and simulation'99 : Proceedings of the eurographics workshop in milano, italy, september 7–8, 1999*. Springer, 1999, p. 133-144 (cf. p. 92).

[Deb+00] Gilles Debunne et al. « Adaptive simulation of soft bodies in real-time ». In : *Proceedings computer animation 2000*. IEEE, 2000, p. 15-20 (cf. p. 93).

[Deb+] Gilles Debunne et al. « Dynamic Real-Time Deformations using Space and Time Adaptive Sampling ». en. In : (), p. 7 (cf. p. 83, 93).

[DGW11] C. Dick, J. Georgii et R. Westermann. « A Hexahedral Multigrid Approach for Simulating Cuts in Deformable Objects ». en. In : *IEEE Transactions on Visualization and Computer Graphics* 17.11 (nov. 2011), p. 1663-1675 (cf. p. 10, 96, 111).

[DBB11] Raphael Diziol, Jan Bender et Daniel Bayer. « Robust real-time deformation of incompressible surface meshes ». In : *Proc. of the ACM Siggraph/Eurographics symposium on computer animation*. 2011, p. 237-246 (cf. p. 80).

[DV21] J. Dupuy et K. Vanhoey. « A Halfedge Refinement Rule for Parallel Catmull-Clark Subdivision ». In : *Computer Graphics Forum* 40.8 (2021). _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14381, p. 57-70. doi : https://doi.org/10.1111/cgf.14381. url : https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14381 (cf. p. 119).

[DSL93] Nira Dyn, Hed. S. et David Levin. *Subdivision schemes for surface interpolation*. Tel Aviv University, 1993 (cf. p. 34).

[Ede87] Herbert Edelsbrunner. *Algorithms in combinatorial geometry*. T. 10. Springer Science & Business Media, 1987 (cf. p. 17).

[Fan+18] Yu Fang et al. « A temporally adaptive material point method with regional time stepping ». In : *Computer graphics forum*. T. 37. Issue : 8. Wiley Online Library, 2018, p. 195-204 (cf. p. 89, 90).

[FSH11] B Fierz, J Spillmann et M Harders. « Element-wise mixed implicit-explicit integration for stable dynamic simulation of deformable objects ». In : *Proceedings of the 2011 ACM SIGGRAPH/ eurographics symposium on computer animation*. 2011, p. 257-266 (cf. p. 89).

[Gan99] Fabio Ganovelli. « Introducing multiresolution representation in deformable object modeling ». In : *Proc. Srpring conf. Computer graphics, comenius univ., 1999*. 1999 (cf. p. 92).

[Gla91] Andrew S Glassner. « Maintaining winged-edge models ». In : *Graphics gems II*. Elsevier, 1991, p. 191-201 (cf. p. 19).

[GB14] Prashant Goswami et Christopher Batty. « Regional time stepping for SPH ». In : *Eurographics 2014*. Eurographics Association, 2014, p. 45-48 (cf. p. 89).

[GKS02] Eitan Grinspun, Petr Krysl et Peter Schröder. « CHARMS : A simple framework for adaptive simulation ». In : *ACM transactions on graphics (TOG)* 21.3 (2002). Publisher : Acm New York, NY, USA, p. 281-290 (cf. p. 95).

[Hop96] Hugues Hoppe. « Progressive meshes ». In : *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 1996, p. 99-108 (cf. p. 26).

[HPH96] Dave Hutchinson, Martin Preston et Terry Hewitt. « Adaptive refinement for mass/spring simulations ». In : *Computer animation and simulation'96 : Proceedings of the eurographics workshop in poitiers, france, august 31–September 1, 1996*. Springer, 1996, p. 31-45 (cf. p. 91).

[Jia+15] Shiyu Jia et al. « CPU–GPU mixed implementation of virtual node method for real-time interactive cutting of deformable objects using OpenCL ». In : *International journal of computer assisted radiology and surgery* 10.9 (2015), p. 1477-1491 (cf. p. 97).

[Jia+18] Shiyu Jia et al. « CPU–GPU Parallel framework for real-time interactive cutting of adaptive octree-based deformable objects ». In : *Computer graphics forum*. T. 37. 2018, p. 45-59 (cf. p. 97).

[Ket99] Lutz Kettner. « Using generic programming for designing a data structure for polyhedral surfaces ». In : *Computational Geometry* 13.1 (1999). Publisher : Elsevier, p. 65-90 (cf. p. 20).

[KE22] Theodore Kim et David Eberle. « Dynamic deformables : implementation and production practicalities ». In : t. ACM SIGGRAPH 2022 Courses. 2022 (cf. p. 77).

[KLB15] Dan Koschier, Sebastian Lipponer et Jan Bender. « Adaptive tetrahedral meshes for brittle fracture simulation ». In : *Proceedings of the ACM SIGGRAPH/ Eurographics symposium on computer animation*. SCA '14. Goslar, DEU : Eurographics Association, 2015, p. 57-66 (cf. p. 93, 96).

[Kos+22] Dan Koschier et al. « A survey on SPH methods in computer graphics ». In : *Computer graphics forum*. T. 41. Number : 2 tex.organization : Wiley Online Library. 2022, p. 737-760 (cf. p. 86).

[Kra08] Pierre Kraemer. « Modèles topologiques pour la multirésolution ». These de doctorat. Strasbourg 1, jan. 2008. url : `https://www.theses.fr/2008STR13117` (visité le 01/02/2023) (cf. p. 37).

[LS07] François Labelle et Jonathan Richard Shewchuk. « Isosurface stuffing : fast tetrahedral meshes with good dihedral angles ». In : *ACM SIGGRAPH 2007 papers*. 2007, 57-es (cf. p. 92).

[MMC16] Miles Macklin, Matthias Müller et Nuttapong Chentanez. « XPBD : position-based simulation of compliant constrained dynamics ». In : *Proceedings of the 9th international conference on motion in games*. 2016, p. 49-54 (cf. p. 81).

[Man+17] P-L Manteaux et al. « Adaptive physically based models in computer graphics ». In : *Computer graphics forum*. T. 36. 2017, p. 312-337 (cf. p. 10, 95, 102).

[Män87] Martti Mäntylä. *An introduction to solid modeling*. Computer Science Press, Inc., 1987 (cf. p. 20).

[MBF04] Neil Molino, Zhaosheng Bao et Ron Fedkiw. « A virtual node algorithm for changing mesh topology during simulation ». In : *ACM Transactions on Graphics (TOG)* 23.3 (2004). Publisher : ACM New York, NY, USA, p. 385-392 (cf. p. 97).

[Mol+03] Neil Molino et al. « A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. » In : *IMR* 103 (2003). Publisher : Citeseer, p. 114 (cf. p. 92).

[Mon92] Joe J Monaghan. « Smoothed particle hydrodynamics ». In : *Annual review of astronomy and astrophysics* 30.1 (1992). Publisher : Annual Reviews 4139 El Camino Way, PO Box 10139, Palo Alto, CA 94303-0139, USA, p. 543-574 (cf. p. 84).

[MG04] Matthias Müller et Markus H Gross. « Interactive virtual materials. » In : *Graphics interface*. T. 4. 2004, p. 239-246 (cf. p. 96).

[Mül+05] Matthias Müller et al. « Meshless deformations based on shape matching ». In : *ACM transactions on graphics (TOG)* 24.3 (2005), p. 471-478 (cf. p. 78).

[Mül+07] Matthias Müller et al. « Position based dynamics ». In : *Journal of Visual Communication and Image Representation* 18.2 (2007), p. 109-118 (cf. p. 81).

[Mül+22] Matthias Müller et al. « Physically based shape matching ». In : *Computer graphics forum*. T. 41. Number : 8 tex.organization : Wiley Online Library. 2022, p. 1-7 (cf. p. 81).

[NSO12] Rahul Narain, Armin Samii et James F O'brien. « Adaptive anisotropic remeshing for cloth simulation ». In : *ACM transactions on graphics (TOG)* 31.6 (2012). Publisher : ACM New York, NY, USA, p. 1-10 (cf. p. 94).

[NFP06] Matthieu Nesme, François Faure et Yohan Payan. « Hierarchical multiresolution finite element model for soft body simulation ». In : *International symposium on biomedical simulation*. tex.organization : Springer. 2006, p. 40-47 (cf. p. 10, 92, 100).

[Nes+09] Matthieu Nesme et al. « Preserving topology and elasticity for embedded deformable models ». In : *ACM transactions on graphics (TOG)*. T. 28. 2009, p. 52 (cf. p. 9).

[NS01] Han-Wen Nienhuys et A Frank van der Stappen. « A surgery simulation supporting cuts and finite element deformation ». In : *International conference on medical image computing and computer-assisted intervention*. 2001, p. 145-152 (cf. p. 96).

[Ota+07] Miguel A Otaduy et al. « Adaptive deformations with fast tight bounds ». In : *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 2007, p. 181-190 (cf. p. 91, 94).

[Pau+15] C. Paulus et al. « Virtual cutting of deformable objects based on efficient topological operations ». In : *Visual Computer* 31.6-8 (juin 2015), p. 831-841 (cf. p. 96).

[Pee+18] Andreas Peer et al. « An implicit SPH formulation for incompressible linearly elastic solids ». In : *Computer graphics forum*. T. 37. Number : 6 tex.organization : Wiley Online Library. 2018, p. 135-148 (cf. p. 86, 88).

[QMD21] Di Qi, Nicholas Milef et Suvranu De. « Divided Voxels : an efficient algorithm for interactive cutting of deformable objects ». In : *The Visual Computer* 37 (2021), p. 1113-1127 (cf. p. 97).

[RJ07] Alec R Rivers et Doug L James. « FastLSM : fast lattice shape matching for robust real-time deformation ». In : *ACM Transactions on Graphics (TOG)* 26.3 (2007), 82-es (cf. p. 9, 80).

[Sei+11] Martin Seiler et al. « Robust interactive cutting based on an adaptive octree simulation mesh ». In : *The Visual Computer* 27.6-8 (2011). tex.ids= seilerRobustInteractiveCutting2011, p. 519-529 (cf. p. 97).

[Sif+07] E Sifakis et al. « Hybrid simulation of deformable solids ». In : *Proceedings of the 2007 ACM SIGGRAPH/ Eurographics symposium on Computer animation*. 2007, p. 81-90 (cf. p. 93).

[SDF07] Eftychios Sifakis, Kevin G Der et Ronald Fedkiw. « Arbitrary cutting of deformable tetrahedralized objects ». In : *Proceedings of the 2007 ACM SIGGRAPH/ Eurographics symposium on Computer animation*. 2007, p. 73-80 (cf. p. 97).

[SG98] Oliver G Staadt et Markus H Gross. *Progressive tetrahedralizations*. IEEE, 1998 (cf. p. 27).

[SOG08] Denis Steinemann, Miguel A Otaduy et Markus Gross. « Fast adaptive shape matching deformations ». In : *Proc. of the ACM Siggraph/Eurographics symposium on computer animation*. 2008, p. 87-94 (cf. p. 80, 97).

[Ste+06] Denis Steinemann et al. « Hybrid cutting of deformable solids ». In : *IEEE virtual reality conference (VR 2006)*. 2006, p. 35-42 (cf. p. 96).

[Ter+87] Demetri Terzopoulos et al. « Elastically deformable models ». In : *ACM Siggraph Computer Graphics* 21.4 (1987), p. 205-214 (cf. p. 9, 69).

[Unt13] Lionel Untereiner. « Representation of multiresolution meshes : an application to subdivision volumes ». fr. Thèse de doct. Université de Strasbourg, nov. 2013. URL : https://theses.hal.science/tel-00951049 (visité le 01/02/2023) (cf. p. 37).

[Unt+15] Lionel Untereiner et al. « CPH : a compact representation for hierarchical meshes generated by primal refinement ». In : *Computer Graphics Forum* 34.8 (2015), p. 155-166 (cf. p. 39, 41, 61).

[VD22] Kenneth Vanhoey et Jonathan Dupuy. « A Halfedge Refinement Rule for Parallel Loop Subdivision ». In : *Eurographics 2022 - Short Papers*. Sous la dir. de Nuria Pelechano et David Vanderhaeghe. ISSN : 1017-4656. The Eurographics Association, 2022. ISBN : 978-3-03868-169-4. DOI : 10.2312/egs.20221028 (cf. p. 119).

[VB05] Julien Villard et Houman Borouchaki. « Adaptive meshing for cloth animation ». In : *Engineering with Computers* 20 (2005). Publisher : Springer, p. 333-341 (cf. p. 91).

[Wic+10] Martin Wicke et al. « Dynamic local remeshing for elastoplastic simulation ». In : *ACM Transactions on graphics (TOG)* 29.4 (2010). Publisher : ACM New York, NY, USA, p. 1-11 (cf. p. 91, 94).

[WT08] Chris Wojtan et Greg Turk. « Fast viscoelastic behavior with thin features ». In : *ACM SIGGRAPH 2008 papers*. 2008, p. 1-8 (cf. p. 92).

[WWD14] Jun Wu, Rüdiger Westermann et Christian Dick. « Real-time haptic cutting of high-resolution soft tissues. » In : *MMVR*. 2014, p. 469-475 (cf. p. 97).

[WWD15] Jun Wu, Rüdiger Westermann et Christian Dick. « A survey of physically based simulation of cuts in deformable bodies ». In : *Computer graphics forum*. T. 34. 2015, p. 161-187 (cf. p. 96).

[Wu+01] Xunlei Wu et al. « Adaptive Nonlinear Finite Elements for Deformable Body Simulation Using Dynamic Progressive Meshes ». en. In : *Computer Graphics Forum* 20.3 (sept. 2001), p. 349-358. ISSN : 0167-7055, 1467-8659. DOI : 10.1111/1467-8659.00527. URL : http://doi.wiley.com/10.1111/1467-8659.00527 (visité le 15/09/2020) (cf. p. 90, 91, 93).

Université de Strasbourg

Quentin WENDLING

Couplage géométrie/mécanique pour l'animation d'objets détaillés

École doctorale Mathématiques, sciences de l'information et de l'ingénieur | ED 269 Université de Strasbourg

Résumé

La thèse propose une approche novatrice pour la représentation et la simulation numérique en temps réel d'objets déformables hautement détaillés en informatique graphique. Nous introduisons le concept de "vues adaptatives" basé sur les cartes combinatoires multirésolution, une représentation multirésolution pour concilier les exigences visuelles élevées avec des performances en temps réel. Cette approche ajuste la résolution des simulations pour une précision accrue dans les zones déformées tout en préservant les performances globales. Étendue aux simulations mécaniques, elle offre une méthode efficace de découpe d'objet et est validée par des benchmarks montrant une amélioration significative tout en maintenant des résultats visuellement similaires. Dans cette recherche, des critères d'adaptation basés sur les interactions objet-environnement sont définis, avec des mécanismes efficaces pour propager les déformations entre les différentes résolutions.

Mots clés : Simulation, Découpes, Modèle topologique

Résumé en anglais

The thesis proposes a novel approach for the real-time representation and numerical simulation of highly detailed deformable objects in graphics computing. We introduce the concept of "adaptive views" based on multi-resolution combinatorial maps, a multi-resolution representation to reconcile high visual requirements with real-time performance. This approach adjusts the resolution of simulations for increased accuracy in deformed areas, while preserving overall performance. Extended to mechanical simulations, it offers an efficient method of object cutting and is validated by benchmarks showing significant improvement while maintaining visually similar results. In this research, adaptation criteria based on object-environment interactions are defined, with effective mechanisms for propagating deformations between different resolutions.

Keywords : Simulation, Cuts, Topological model