HAL
open science

Visualisation distante temps-réel de grands volumes de données

Sébastien Barbier

To cite this version:

Sébastien Barbier. Visualisation distante temps-réel de grands volumes de données. Modélisation et simulation. Université Joseph-Fourier - Grenoble I, 2009. Français. NNT : . tel-00438114

HAL Id: tel-00438114

https://theses.hal.science/tel-00438114v1

Submitted on 2 Dec 2009

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

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

Université Grenoble I - Joseph Fourier
École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique

Visualisation Distante Temps-Réel de Grands Volumes de Données

Sébastien BARBIER

Thèse
présentée pour l'obtention du titre de
Docteur de l'Université Joseph Fourier
Spécialité Informatique

Arrêté ministériel du 30 mars 1992 et du 7 août 2006
Préparée au sein du
Laboratoire Jean Kuntzmann (LJK), Projet EVASION, UMR CNRS 5224.

Dirigée par Professeur Georges-Pierre BONNEAU

Soutenue le 26 octobre 2009

Composition du jury :

Jean-Michel	DISCHLER	Rapporteur
Wilfrid	LEFER	Rapporteur
Marie-Paule	CANI	Examinatrice
Guillaume	THIBAULT	Examinateur
Georges-Pierre	BONNEAU	Examinateur

Remerciements

Une thèse, c'est un peu comme une épopée. Il y a un fil conducteur, une essence même à l'histoire, un titre, une idée scientifique, des objectifs. Mais finalement, ce qui en fait la force, ce sont les personnages qui y vivent, qui viennent et qui repartent. Qui, grâce à leur propre façon de voir les choses, font avancer la réflexion. Il est donc plus que capital de remercier tous ces personnages qui alimentent cette épopée. Notre façon de communiquer nous impose pourtant d'écrire les choses dans un certain ordre. Cet ordre n'a pas lieu d'être ici. Chaque personne est importante comme chaque pierre a son propre rôle dans une montagne. On pourrait ainsi lire les lignes qui vont suivre dans l'ordre ou le désordre. Il est possible, la mémoire jouant parfois d'espièglerie, que l'un d'entre eux soit oublié au moment de l'écrire sur le papier. Qu'il m'en excuse.

Je tiens à remercier mon père et ma mère qui ont su placer une confiance inébranlable en moi et qui m'ont permis de mener à bien ces études supérieures jusqu'ici et être toujours présents dans les moments faciles comme les difficiles. Un énorme merci à l'ensemble de ma famille qui a toujours été à mes côtés. Votre présence et votre soutien m'ont toujours été des biens précieux.

Je souhaite remercier mon directeur de thèse, Georges-Pierre Bonneau, qui a su me faire découvrir la visualisation scientifique, me donner la passion de la recherche, aiguiser mon esprit critique, motiver ma curiosité. Sa bonne humeur comme son professionnalisme ont été des véritables moteurs et ce fut une bien belle thèse passée à ses côtés.

Vient la foule, une équipe de vie au jour le jour dont il sera difficile de s'évader. Merci à vous tous qui avait rempli ses longues journées de soumission, d'implantation, de résultats, de débogage... dans un esprit de collaboration et de camaraderie si bien incarné par notre chef d'équipe Marie-Paule Cani. Merci donc à vous Franck, François, Éric, Lionel, Fabrice, Olivier, Jean-Claude, Stéphane, Florence, Mathieu, Midori, Matthieu, Antoine, Julien, Cécile, Sahar, Damien, Adeline, Maxime, Cyril, Guillaume, Noura, Romain, Alexandre, Laurent, André, Lenka, Benjamin, Paul, Qizhi, LuLin, Grégoire, Cédric, Yacine, Jean-Rémy, François, Antonin, Toinou, Bérangère, Adyl, et ceux que j'aurais pu oublier. Il fut bon vivre à côté de vous chaque jour durant, de jouer aux cartes, d'apprendre le football, de partager votre bureau, de converser de sciences comme de tout et rien. Merci aussi à l'ensemble de l'équipe Artis pour leur camaraderie.

Merci à toi qui s'est si bien occupé de nous pendant toutes ces années. Qui était là pour nous accueillir, avec ce sourire et ce rire qui étaient comme des rayons de soleil. Merci Anne.

Merci à toutes ces personnes que j'ai croisées une heure, un jour, une semaine. Avec qui il fut tant plaisant de converser : Julien, Pierre, Mélanie, Adrien, Thierry, Stefanie, Basile... Et aux personnes qui ont su m'accompagner lors de mon séjour à Gênes : *gracie mile Ana, Daniela e Donatella* ainsi qu'à mon équipe d'accueil génoise G^3 dont *la professore Leila de Floriani*. Enfin, à ceux qui ont été là comme autant de points d'exclama-

tion dans une phrase contenant des instants de joie : Benjamin, Guillaume, Stéphane, Édouard, Mickaël, David et Delphine...

C'est là qu'apparaissent les joyeux drilles, les fanfarons, la troupe baptisée *SOFA team*. Il y a tant de bons souvenirs... tant de soirées joviales, de vacances ensoleillées, d'éclats de rire. Comme autant de cartes postales qui ravissent l'esprit. Merci pour votre bonne humeur, votre patience, votre gentillesse. Il ne fut pas aisé de vivre quotidiennement avec moi et vous vous en êtes sortis avec brio. Merci donc Tonton, Papa, Pof et Marie.

Viennent aussi ceux qui me supportent depuis tant de temps. Et qui sont encore là malgré les distances et dont l'amitié a su me guider pendant ces trois années. Merci à vous mes amis, irremplaçables dans mes yeux et dans mon cœur : Émilie, Fanny, Céline, Mikaël, Nelly, Delinger, Ariane, Vincent, Hadrien, Romain, Rémi ainsi qu'à mes deux irremplaçables amis de Champollion Jérémy et Nicolas dont les repas festifs passés ensemble furent comme les gerbes dorés d'un feu d'artifice ! À vous aussi, mes camarades de promotion : Sébastien, Olivier, Vincent, Jérôme, Jean-Denis, Chama... Que de moments agréables comme le parfum d'une fleur sous le doux alizé...

Surtout il ne faut pas oublier le brillant ingénieur, Thomas, qui a partagé pendant six mois la route de cette thèse, qui a su y apporter tant de choses et dont l'amitié et l'estime me sont chères. Merci TomTom par delà les océans !

Puis, il y a le musicien, le rire, l'humour et la camaraderie. Quelques mots ne suffiront pas pour t'exprimer ma gratitude et pour dire qu'il est bon de te connaître et d'avoir croisé ta route. *Tudo bem ?* Merci *Gros* !

Reste l'irremplaçable, celui qui a partagé un bon bout de route avec moi bien avant déjà cette épopée, qui a su fournir écoute et patience, rire et autodérision, qui dans les moments difficiles à éclairer les pavés de sa culture, de ses convictions inébranlables et parfois si proche de la vérité. Avec qui j'ai vogué sans jamais chavirer. Merci Pierre-Édouard, merci !

Enfin, je tenais à remercier les trois frères qui ont guidé mes pas dans la vie comme dans la science.

Tout d'abord mon frère de science dont les conseils, l'écoute et la sagesse ont su m'apporter tant de choses. Qu'il fut bon de converser, d'inventer, d'imaginer à tes côtés. De refaire le monde. De rire ou de se taire. Simplement de partager. Que la route fut belle et espérons qu'elle le soit encore longuement. Merci Christian.

Puis mes frères de cœur, ceux avec qui j'ai partagé un toit pendant trois ans, qui ont supporté mes manies, mes colères, mes fatigues, mes mauvaises blagues. Qui ont toujours été là avec leurs qualités comme leurs petits défauts, leur grande générosité, leur humour quotidien, leur simplicité. Cette deuxième famille pour moi, la famille Hars. Merci Hugo, merci Axel.

Enfin, mon petit frère, Yvan, qui m'a appris beaucoup de choses pendant cette grande aventure : la ténacité, le courage, l'humilité. Tu restes irremplaçable.

Résumé

La simulation numérique génère des maillages de plus en plus gros pouvant atteindre plusieurs dizaines de millions de tétraèdres. Ces ensembles doivent être visuellement analysés afin d'acquérir des connaissances relatives aux données physiques simulées pour l'élaboration de conclusions. Les capacités de calcul utilisées pour la visualisation scientifique de telles données sont souvent inférieures à celles mises en œuvre pour les simulations numériques. L'exploration visuelle de ces ensembles massifs est ainsi difficilement interactive sur les stations de travail usuelles. Au sein de ce mémoire, nous proposons une nouvelle approche interactive pour l'exploration visuelle de maillages tétraédriques massifs pouvant atteindre plus de quarante millions de cellules. Elle s'inscrit pleinement dans le procédé de génération des simulations numériques, reposant sur deux maillages à résolution différente – un fin et un grossier – d'une même simulation. Une partition des sommets fins est extraite guidée par le maillage grossier permettant la reconstruction à la volée d'un maillage dit *birésolution*, mélange des deux résolutions initiales, à l'instar des méthodes multirésolution usuelles. L'implantation de cette extraction est détaillée au sein d'un processeur central, des nouvelles générations de cartes graphiques et en mémoire externe. Elles permettent d'obtenir des taux d'extraction inégalés par les précédentes approches. Afin de visualiser ce maillage, un nouvel algorithme de rendu volumique direct implanté entièrement sur carte graphique est proposé. Un certain nombre d'approximations sont réalisées et évaluées afin de garantir un affichage interactif des maillages birésolution.

Abstract

Numerical simulations produce huger and huger meshes that can reach dozens of million tetrahedra. These datasets must be visually analyzed to understand the physical simulated phenomenon and draw conclusions. The computational power for scientific visualization of such datasets is often smaller than for numerical simulation. As a consequence, interactive exploration of massive meshes is barely achieved. In this document, we propose a new interactive method to interactively explore massive tetrahedral meshes with over forty million tetrahedra. This method is fully integrated into the simulation process, based on two meshes at different resolutions – one fine mesh and one coarse mesh – of the same simulation. A partition of the fine vertices is computed guided by the coarse mesh. It allows the on-the-fly extraction of a mesh, called *biresolution*, mixed of the two initial resolutions as in usual multiresolution approaches. The extraction of such meshes is carried out into the main memory (CPU), the last generation of graphics cards (GPU) and with an out-of-core algorithm. They guarantee extraction rates never reached in previous work. To visualize the biresolution meshes, a new direct volume rendering (DVR) algorithm is fully implemented into graphics cards. Approximations can be performed and are evaluated in order to guarantee an interactive rendering of any biresolution meshes.

SOMMAIRE

Introduction 11

1 Visualiser des Maillages Tétraédriques : Motivations et Enjeux 15

2 Bi-Résolution : Concepts et Précalculs 37

3 Bi-Résolution : Extraction Dynamique 65

4 Visualiser via le Rendu Volumique 89

Conclusion 119

A Ensembles de Données 125

B Temps de Précalculs 129

Table des matières 131

Table des figures 135

Index 138

Bibliographie 141

Introduction

Le secret de toute méthode consiste à regarder avec soin dans toute chose ce qu'il y a de plus absolu.

René DESCARTES - *Règle pour la direction de l'esprit*

Comprendre l'Univers et les lois intangibles qui le régissent, valider des innovations technologiques et scientifiques, observer le corps humain comme les interactions atomiques, prédire le temps ou les catastrophes naturelles sont les aspirations de l'humanité depuis que la science est née en Mésopotamie vers $-3\,500$ ([Pic91], p.3). À l'ère de la révolution informatique (ou troisième révolution industrielle), ces aspirations prennent une nouvelle dimension grâce à la *simulation numérique* et à la *visualisation scientifique*.

La *simulation numérique* reproduit des phénomènes réels sur la base de modèles théoriques dans le but de prédire des événements (météorologie, astronomie, géologie, ...) ou infirmer, valider des expériences (thermodynamique, neutronique, aérodynamique, ...). Elle utilise une discrétisation de l'espace en grilles régulières ou irrégulières pour appliquer les méthodes dites des éléments finis volumiques afin de résoudre numériquement les problèmes initialement théoriques. Un ensemble de résultats numériques est ainsi obtenu, représentatif de l'état du système à un certain pas de temps. Ces résultats sont alors exploitables en tant que données brutes.

La *visualisation scientifique* permet de représenter grâce à des techniques adaptées à chaque domaine scientifique les résultats de ces simulations numériques. Cette représentation est assurée par la création d'images générées par ordinateur. Ces images sont produites dans le but d'exploiter puis de diffuser les résultats à l'instar des dessins industriels (initiés dans leur forme contemporaine par Gaspard Monge au XIXème siècle) ou des dessins anatomiques (comme ceux de *De humani corporis fabrica libri septem* écrit en 1543 par Andreas Vesalius). Ainsi, la visualisation scientifique permet de mettre en évidence les tendances des phénomènes simulés et d'appréhender des comportements inattendus. Puis dans un second temps, elle assure un rôle de communication en permettant la diffusion des résultats. Il est ainsi devenu presque familier de voir de telles images et vidéos dans les magazines de vulgarisation scientifique pour illustrer les résultats d'une simulation en astronomie[1] ou en écoulement de fluides[2].

Au cours de l'exploitation, l'*interactivité* de telles représentations permet à l'utilisateur de modifier le point de vue comme s'il tournait ou se rapprochait de l'objet et ainsi d'acquérir une vision tridimensionnelle de l'es-

[1] On peut citer la vidéo produite pour la simulation issue des observations du *6df Galaxy Survey* : http ://www.aao.gov.au/local/www/6df/movies/6df_spiral_cd.mov.

[2] On peut citer les images et vidéos diffusées par le studio de visualisation scientifique de la NASA sur les acquisitions au sein de cyclones : http ://www.nasa.gov/vision/earth/lookingatearth/rita_hot_towers.html.

pace et des données à travers la création des images bidimensionnelles. Cette interactivité est ainsi une clef essentielle pour permettre l'exploitation visuelle des résultats.

À l'heure où la création de *grosses masses de données* est facilitée par l'évolution continuelle de la puissance de calcul des supercalculateurs (depuis les 20 opérations flottantes à la seconde du *Z3* en 1941 aux quelques milles pétaflops[3] du *RoadRunner* en 2008), l'interactivité des techniques de visualisation est toujours un prérequis essentiel pour exploiter les données sur des machines qui se révèlent souvent moins performantes. Un fossé de performances est ainsi sans cesse entretenu entre la taille gigantesque des données simulées et les capacités de calculs permettant leur visualisation.

Il semble donc fondamental de développer un ensemble de solutions permettant de garantir la visualisation interactive de tels grands volumes de données.

Contexte et Motivations

En 2007, avec la création du Grand Équipement National de Calcul Intensif (*GENCI*), la France s'impose comme *leader* européen dans les domaines de la modélisation, de la simulation et du calcul intensif. Il en résulte la mise en place de supercalculateurs puissants : au Centre Commun de Recherche et Technologie (*CCRT*) avec une puissance de calculs de $103 + 192$ téraflops et une mémoire vive de 25 téraoctets ; et au Centre Informatique National de l'Enseignement Supérieur (*CINES*). Cette puissance de calculs permet ainsi la génération de données massives issues entre autres de simulations numériques.

Par exemple, dans le cadre des grilles irrégulières, des maillages composés de plus de quarante millions de tétraèdres peuvent être générés pour des études de thermodynamique au sein de chambres de combustion ou de plus de quinze millions pour des études d'élastodynamique. La communauté des géologues produit aussi des maillages tétraédriques massifs pour les études pétrographiques, pétrologiques ou sismiques.

De telles données massives ne peuvent être exploitées dans leur globalité par la visualisation scientifique pour des raisons principalement matérielles. Des techniques de prétraitement des données ont été ainsi élaborées pour garantir une visualisation interactive en reposant sur le fait que l'être humain ne peut interpréter qu'une quantité d'information visuelle restreinte à un instant donné. Les approches *multirésolution* proposent une exploitation pertinente des données en se basant sur une hiérarchie de représentations de plus en plus simplifiées des données initiales. Ces *niveaux de détails* sont extraits en préservant les caractéristiques des champs simulés dans l'optique d'être combinés pour adapter la résolution des données en fonction des nécessités de l'utilisateur. Ainsi, dynamiquement, des régions à haute résolution sont mélangées à des régions à basse résolution garantissant la conservation des données initiales dans les espaces d'intérêts définis par l'utilisateur.

Lors de l'exploitation des résultats, ces techniques permettent de réduire le nombre de primitives géométriques sur lesquelles ont été calculées les simulations. Les algorithmes de visualisation sont alors déployés sur une quantité d'information moindre par rapport aux données initiales. Leur implantation efficiente permet de garantir la production d'images à un taux de rafraîchissement interactif. La manipulation des données est ainsi facilitée afin de comprendre les structures tridimensionnelles, les relations d'occlusions et les tendances comme les singularités des résultats.

Néanmoins, de telles solutions pour l'exploitation des grilles irrégulières restent encore limitées car elles se révèlent consommatrices en mémoire et chronophages en précalculs malgré les améliorations matérielles. Elles n'assurent pas des vitesses d'extraction suffisantes pour garantir une exploitation interactive des données pour l'ensemble des techniques de visualisation même lorsque celles-ci sont implantées de manière efficiente.

Il est ainsi impossible de prétraiter rapidement puis de visualiser interactivement de gros maillages tétraédriques d'une dizaine de millions de tétraèdres, maillages dont la taille est aujourd'hui usuelle dans de nombreux domaines scientifiques ayant recours à la simulation numérique.

[3] giga = 10^9, tera = 10^{12}, péta = 10^{15}

Contributions

Les contributions proposées au sein de cette thèse poursuivent ainsi un même objectif : celui de faciliter l'exploitation visuelle de maillages tétraédriques massifs issus de simulations numériques grâce à un ordinateur de bureau. Pour cela, nos efforts se sont portés sur quatre points essentiels :

L'accélération du prétraitement des données. L'exploitation des données massives ne peut être effectuée interactivement sans leur transformation préalable afin d'en simplifier une partie à l'instar des approches multirésolution. Nous proposons des nouveaux algorithmes de précalculs assurant à la fois une faible consommation mémoire et une faible complexité temporelle pour l'élaboration d'un schéma reposant uniquement sur deux niveaux de détails (et non une séquence finie comme les approches multirésolution usuelles). Ces méthodes permettent d'assurer le traitement de maillages tétraédriques composés de plus de quarante millions de tétraèdres au sein de deux gigaoctets de mémoire vive.

L'accélération de la création de niveaux de détails dynamiques. La création des niveaux de détails dynamiques pour les maillages tétraédriques dans le cadre de méthodes multirésolution usuelles reposent sur des structures hiérarchiques coûteuses à mettre à jour. *Via* l'utilisation de deux niveaux de détails statiques, nous simplifions la construction de tels niveaux de détails et garantissons des vitesses d'extraction jamais atteintes. L'absence de structures de données complexes de mise-à-jour permet aussi le portage d'une telle méthode sur la carte graphique afin de bénéficier de la totalité de sa puissance de calculs.

L'exploitation interactive de grandes masses de données. La taille des données toujours plus massives représente un défi supplémentaire lors de l'élaboration d'une solution interactive pour l'exploitation de données. Les limitations mémorielles des ordinateurs de bureau et des cartes graphiques sont le principal frein. Nous prenons en compte de telles limitations et proposons une implantation en mémoire externe afin de s'en affranchir.

L'interactivité de techniques de visualisation pour les schémas multirésolution. Les techniques de visualisation font habituellement partie intégrante de l'approche multirésolution : on parle d'un rendu multirésolution. Mais elles n'ont jusqu'alors jamais été adaptées aux schémas multirésolution de manière indépendante c'est-à-dire de façon à considérer les modifications dynamiques de connectivité et de géométrie du maillage affiché à l'écran. Nous adaptons des techniques préexistantes de rendu volumique direct ordonné afin que celles-ci soient compatibles avec les niveaux de détails dynamiques et afin de garantir une manipulation interactive des données.

L'ensemble de ces améliorations permet d'assurer la visualisation interactive de grands ensembles de données tétraédriques.

Organisation du document

Le premier chapitre de ce mémoire décrit le contexte actuel de la visualisation scientifique tant dans ses objectifs, ses performances que dans ses enjeux. Nous soulignons la place essentielle de la visualisation scientifique au sein d'une boucle de traitement permettant l'exploitation des résultats issus de la simulation. Nous spécialisons notre approche sur les maillages tétraédriques et dressons une vue d'ensemble des techniques de création et de visualisation pour de tels maillages. Nous discutons ensuite des améliorations architecturales des cartes graphiques et des nouvelles possibilités qu'elles offrent avant de mettre en emphase les différents enjeux auxquels est confrontée la visualisation scientifique en cette fin des années 2000.

Le deuxième chapitre présente les précédentes approches de simplification et multirésolution pour les maillages tétraédriques afin de répondre au traitement des grosses masses de données. Il introduit ensuite les prétraitements nécessaires à la création de notre propre solution reposant sur l'extraction d'un maillage birésolution. Nous démontrons que notre approche permet de traiter en mémoire vive des maillages plus

conséquent en taille et plus rapidement que les prétraitements liés aux précédents algorithmes de simplification et multirésolution. Nous démontrons aussi, sous certains conditions, la généralité de notre approche.

Le troisième chapitre détaille l'extraction du maillage birésolution d'une manière formelle puis sous la forme de trois implantations : au sein du processeur central, de la carte graphique et grâce à un traitement en mémoire externe. Nous démontrons que ce nouveau schéma d'extraction permet d'atteindre des vitesses d'extraction de niveaux de détails jusqu'alors inégalées par les précédentes approches et de traiter de gros ensembles de données.

Le quatrième chapitre s'intéresse à l'intégration des techniques de visualisation au sein de notre schéma birésolution. Nous nous intéressons plus particulièrement à l'adaptation d'algorithmes de rendu volumique direct afin d'obtenir des temps d'exploration interactifs. Nous revenons sur les techniques préexistantes et proposons un nouvel algorithme de rendu volumique adapté aux approches multirésolution. Nous détaillons aussi l'intégration des autres techniques de visualisation à notre schéma birésolution.

Enfin, nous concluons en évoquant les principaux défis que devra relever la visualisation scientifique afin de répondre pleinement aux attentes des utilisateurs au cours des prochaines années et de se placer à la croisée de nombreux domaines scientifiques pour en faire rejaillir la quintessence.

CHAPITRE 1

Visualiser des Maillages Tétraédriques : Motivations et Enjeux

> The ability to scientists to visualize calculations and complex simulations is absolutely essential to ensure the integrity of analyses, to promote scrutiny in depth and communicate the result of such scrutiny to others... The purpose of scientific calculation is looking, not enumerating.
>
> Bruce H. McCormick - *Visualization in Scientific Computing*, 1987

La simulation numérique est devenue incontournable dans de nombreux domaines scientifiques comme la biologie, la chimie, l'hydrostatique, la thermodynamique, la géologie, l'aérodynamique, l'astronomie, la neutronique ou l'électromagnétique. En effet, l'ordinateur de part sa puissance de calcul et sa maniabilité est devenu l'outil indispensable pour valider des hypothèses plus ou moins complexes dans le cadre d'une étude scientifique. Cette solution qui vient en complément à la mise en place de modèles réduits concrets et onéreux et qui repose sur des axiomes mathématiques et physiques solides s'est ainsi imposée au cours des deux dernières décennies.

La réalisation de telles simulations produit une quantité de données chiffrées qu'un ingénieur ou un chercheur doit par la suite analyser afin d'en tirer des conclusions. La compréhension d'une telle masse de données est impossible en l'état brut et des représentations alternatives de ces résultats sont ainsi nécessaires afin d'aboutir à un raisonnement et par extension à l'extraction de connaissances permettant la (contre-) validation de la simulation.

La visualisation scientifique s'inscrit dans l'optique de proposer des méthodes ayant pour but d'aider à la compréhension puis à la diffusion de tels résultats via l'extraction d'images porteuses de sens à partir des données brutes. Son rôle est donc vitale pour valider les simulations numériques dans le sens où elle permet de créer un retour visuel concret comme le permet la simulation sur modèles réduits.

À l'ère où les progrès de l'électronique permettent la création de processeurs toujours plus puissants et

spécialisés pour la simulation et le graphisme, la visualisation scientifique, discipline récente [1], se trouve confronter sans cesse à de nouveaux défis tant sur le traitement des données que sur leur représentation.

Ce chapitre a pour but de dresser un tour d'horizon de la place de la visualisation scientifique au sein de la communauté scientifique (section 1), des bases théoriques permettant la création de maillages tétraédriques utiles à la simulation (section 2), des différentes techniques de représentations des données issues des simulations (section 3) et de l'avènement des cartes graphiques (section 4) pour mettre en évidence les enjeux et les possibilités s'offrant à ce domaine vingt ans après sa naissance (section 5).

1 Motivations

La compréhension d'un phénomène au sein d'un environnement naturel ou industriel intervient dans le cadre d'une boucle de découverte scientifique où la puissance de calcul des ordinateurs et leur capacité à produire des images jouent aujourd'hui un rôle déterminant dans des domaines aussi variées que la météorologie, la neutronique, les nanotechnologies ou l'astronomie. En effet, afin de répondre à une problématique précise et apporter un ensemble de solutions fiables, une étude scientifique peut s'inscrire dans un processus composé de trois étapes essentielles : la discrétisation de l'espace d'étude, la mise en place d'une simulation numérique et l'exploitation, notamment visuelle, des résultats. La figure 1.1 illustre le principe de la boucle de la découverte scientifique.

Dans un premier temps, l'espace est discrétisé *via* la création manuelle ou automatique d'un maillage bi- ou tridimensionnel. Ce nouvel espace physique constitue le support de la simulation numérique pouvant reposer sur plusieurs pas de temps. La prise en compte des paramètres physiques et des conditions limites au bord permet de générer un ensemble de résultats associés au maillage. Ces résultats sont ensuite analysés grâce à la création d'images porteuses de sens qui les valident ou les réfutent.

Dans le cas d'invalidité des résultats, la boucle est recommencée soit au niveau de la discrétisation – par exemple, si la granularité des cellules n'est pas assez fine pour garantir la convergence des résultats ; soit au niveau de la simulation – en modifiant par exemple les conditions de bords ou les algorithmes de calcul.

Cette boucle de la découverte scientifique est ainsi dépendante de l'intervention de l'utilisateur. Celui-ci, grâce à son expérience et ses connaissances, agit au niveau de toutes les étapes afin de faciliter l'obtention des résultats, leur exploitation puis leur diffusion. Ainsi, l'utilisateur reste le point central de cette boucle et ne peut en être abstrait : il est donc essentiel de ne pas l'oublier lors de l'élaboration d'algorithmes permettant d'automatiser une partie, voire l'ensemble, de cette boucle.

Dans la suite de cette section, nous détaillons plus précisément chacune de ces trois étapes essentielles : la discrétisation de l'espace nécessaire à la formulation du problème analytique en section 1.1 ; la simulation numérique permettant *via* des équations discrètes de modéliser un phénomène ciblé en section 1.2 et enfin la visualisation indispensable pour l'exploitation et la diffusion des résultats en section 1.3.

1.1 Discrétisation de l'espace

Afin de résoudre les équations mathématiques associées à la simulation, le domaine $\Omega \subset \mathbb{R}^3 \times \mathbb{R}$ doit être discrétisé en un maillage composé d'un certain nombre de cellules. La forme des cellules de cet espace est importante car les modes opératoires, la stabilité et la convergence des calculs en dépendront. La granularité du maillage est tout aussi cruciale, définissant la précision spatiale des calculs. Elle doit donc être fixée avec soin afin que des phénomènes locaux puissent être représentés avec assez de détails pour être repérés et compréhensibles lors de la phase d'exploitation des résultats. L'intervention de l'utilisateur (expert) est ainsi

[1] **Bref Historique :**
Le terme de *Visualisation en Calcul Scientifique* apparaît pour la première fois en 1986 [PT94] au cours d'un meeting organisé par la *National Science Foundation*. Un an plus tard, en 1987, lors du premier workshop spécifique au domaine, McCormick donne la première définition de la Visualisation Scientifique : "*The ability to scientists to visualize calculations and complex simulations is absolutely essential to ensure the integrity of analyses, to promote scrutiny in depth and communicate the result of such scrutiny to others...*" [MC87]. Les années 1990 verront la visualisation scientifique prendre enfin son essor.

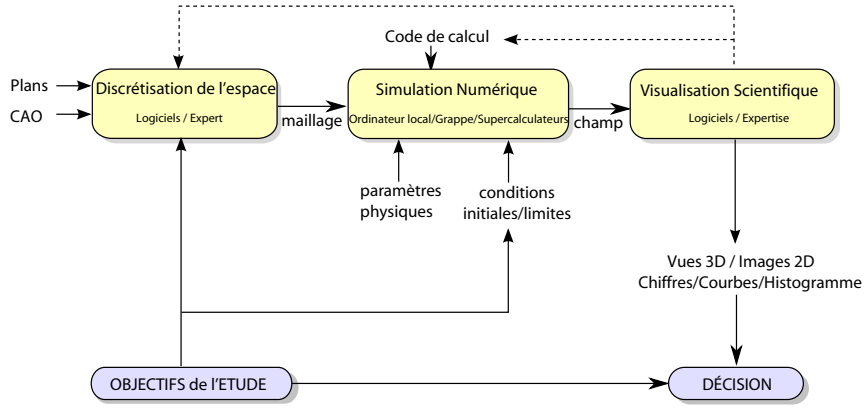


FIG. 1.1: **Boucle de la découverte scientifique** inspirée de [Bou09]. Elle est composée de trois étapes distinctes : la discrétisation de l'espace, la simulation numérique et la visualisation scientifique. Ces étapes s'appuient fortement sur les objectifs d'une étude et permettent de prendre des décisions suite aux résultats obtenus.

primordiale lors de la génération car celle-ci ne peut être automatisée de manière efficace à l'ensemble des simulations possibles (et ainsi des cas d'étude possibles).

Pour les maillages non structurés volumiques, quatre types de cellules sont principalement utilisées : le *tétraèdre*, la *pyramide* (à base carrée), le *prisme* et l'*hexaèdre*. La forme de ces cellules est illustrée au sein de la figure 1.2. On parlera respectivement de maillages tétraédriques, pentaédriques ou hexaédriques si ceux-ci ne sont composés uniquement que de tétraèdres, pyramides ou prismes et hexaèdres respectivement. Lorsque les hexaèdres sont des cubes (ou *voxel* pour *Volume Element*), le maillage est appelé *grille régulière*.

Les grilles hexaédriques régulières assurent orthogonalité et facilité de partitionnement alors que les grilles irrégulières permettent de représenter facilement des topologies complexes. Néanmoins, ces dernières utilisent plus de cellules (quatre à dix fois plus [CA92]) que les premières pour représenter la même simulation. Au contraire, la génération de maillages hexaédriques prend plus de temps que celle de maillages tétraédriques pour des géométries complexes [SJ08]. Face aux avantages et aux inconvénients des deux types de grilles, le maillage créé pour certaines simulations est souvent un *maillage hybride*, mélange d'hexaèdres (ou de primes) et de tétraèdres [KSS08].

En effet, les hexaèdres sont utiles pour modéliser des domaines plastique ou hautement élastique, ou encore les flux laminaires alors que les tétraèdres sont adaptés pour les flux turbulents ou la dynamique des cellules microbiologiques. En dynamique des fluides (ou *CFD* pour *Computational Fluid Dynamics*), des maillages hybrides sont ainsi utilisés composés de tétraèdres au centre des écoulements et de prismes sur les bords des parois. La figure 1.3 illustre la création d'un tel maillage autour d'une soupape d'admission[2].

Les maillages irréguliers sont créés en fonction des besoins selon plusieurs procédés. Pour les maillages tétraédriques, les tétraédralisation dites de Delaunay [She98] ou l'extraction par front avançant [Loh96] (*advancing front*) sont les plus utilisées. Elles nécessitent la connaissance préalable d'un maillage surfacique de bord. Dans certains domaines, l'utilisateur peut ne définir que des contours ou un modèle CAO (Conception Assistée par Ordinateur). Le maillage volumique est donc obtenu en deux temps via l'extraction en premier lieu de surfaces triangulées comme illustré au sein de la figure 1.5. Cet exemple a été généré grâce à un logiciel générateur de maillages d'éléments finis : *GMSH*[3] [GR09]. La conception de maillages tétraédriques *via* les critères de Delaunay est expliquée en détails au sein de la section 2.2.2.

Pour les maillages hybrides utilisés en mécanique des fluides, le maillage tétraédrique est souvent extrait une fois que les prismes le long des bords ont été générés et raffinés. Là encore, ceux-ci sont obtenus via une surface triangulée préalablement fournie [KKM96].

Pour les maillages hexaédriques, de nombreuses techniques existent [Owe98] mais sont rarement robustes et généralisables dans toutes les situations. L'utilisateur doit donc souvent prédécouper la géométrie de base en

[2]Ce maillage a été généré grâce au logiciel *Engrid* disponible à l'adresse suivante : http ://sourceforge.net/projets/engrid.
[3]disponible à l'adresse suivante : http ://geuz.org/gmsh/

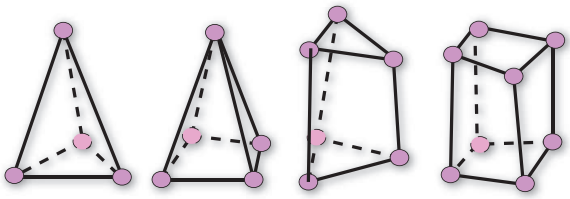


FIG. 1.2: **Cellules pouvant composée un maillage volumique**. De gauche à droite : le tétraèdre (4 sommets, 4 faces triangulaires), les pentaèdres : la pyramide (5 sommets, 4 faces triangulaires + 1 face quadrilatérale) et le prisme (6 sommets, 2 faces triangulaires + 3 faces quadrilatérales), l'hexaèdre (8 sommets, 6 faces quadrilatérales).

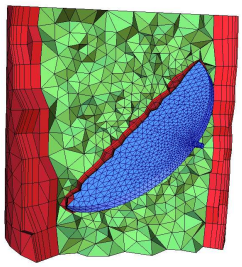


FIG. 1.3: **Maillage hybride pour la simulation de fluides**. Afin d'étudier l'influence d'une soupape d'admission (en bleu) à l'entrée d'un moteur, une simulation est réalisée dans le conduit d'arrivée. Néanmoins, à cause de la viscosité du fluide, un maillage hybride est plus adapté à la résolution des équations. Ainsi, des prismes (en rouge) sont créés au bord des surfaces pour leur régularité alors que des tétraèdres (en vert) sont utilisés au centre. Le maillage a été obtenu via l'utilisation du logiciel *Engrid.*

.

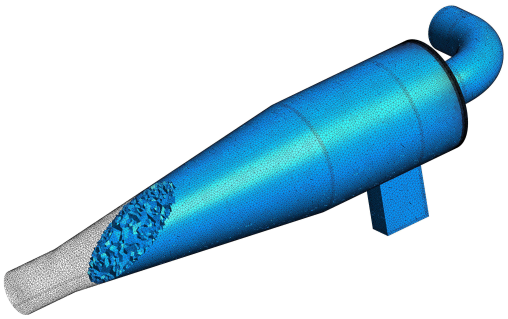


FIG. 1.4: **Maillage hexaédrique pour la simulation de gaz** (Fourni par John F. Porter). Lors de l'évaluation de la faisabilité de la construction d'une centrale électrique basée sur la combustion des déchets, le *cyclone* séparateur entre le gaz et les particules néfastes qu'il véhicule est étudié via la simulation de la loi de Stokes. Le maillage va permettre de valider le comportement du gaz sale arrivant par l'entrée rectangulaire, son échappement par le tuyau incurvé alors que les particules seront rejetées par l'autre extrémité [CPHM04]. Il a été généré via l'outil *GMSH.*

un certain nombre de cubes qui seront ensuite subdivisés afin de générer des hexaèdres. Pour passer outre ce travail fastidieux, une approche topologique, de complexité linéaire, a été développée reposant sur la dualité des maillages hexaédriques. Mais, à l'heure actuelle, cette solution n'a jamais été implantée algorithmiquement. Récemment, l'étude approfondie des contraintes nécessaires à la génération de maillages hexaédriques apporte certaines généralisations dans l'optique d'améliorer les techniques existantes [SJ08]. La figure 1.4 présente un maillage hexaédrique.

Le lecteur intéressé peut se référer à [Owe98] pour plus de détails sur la génération des maillages non structurés. Notons que la visualisation intervient à la fin de cette première étape, lors de l'affichage des maillages générés afin d'en valider la granularité comme la forme comme l'illustre les figures 1.3 et 1.4.

1.2 Simulation

Une fois l'espace discrétisé en cellules adaptées aux équations de la simulation, celle-ci doit être réalisée. Pour cela, les paramètres spécifiques au problème doivent être connus : constantes physiques des matériaux, conditions initiales et limites, pas d'intégration, *etc*... Par exemple, pour la simulation d'écoulement de fluides, les équations de Navier-Stokes, généralement utilisées [GR86], nécessitent entre autres la connaissance de la masse volumique du fluide et de sa viscosité. De plus, un certain nombre d'approximations peut être réalisé influençant les méthodes de résolution et les conditions initiales et limites. Pour les transfert d'énergie [IDBL06], la conductivité thermique du matériau doit être connue ainsi que le modèle appliqué pour modéliser les échanges

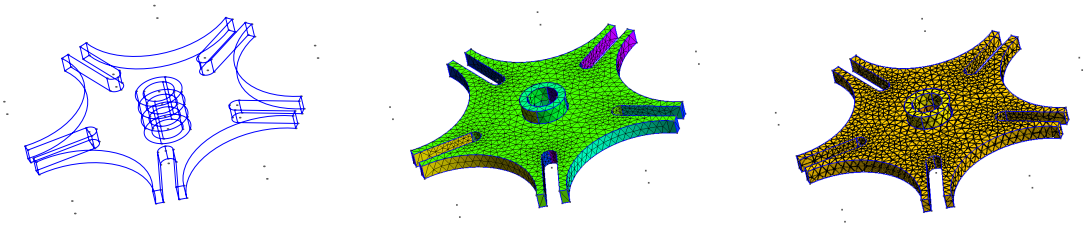


FIG. 1.5: **Exemple des étapes de création d'un maillage tétraédrique pour une pièce mécanique**. De gauche à droite : le squelette de la pièce basée sur une géométrie simple (points, arcs de cercle, segments, ...), la création d'un maillage surfacique sur les bords, la création d'un maillage CDT (cf. définition 1.17) afin de réaliser une simulation physique. Cet ensemble est généré grâce au logiciel *GMSH*.

de transfert aux limites : conditions de Dirichlet, de Neumann ou de Robin-Fourier...

Ces paramètres peuvent aussi résulter de calculs effectués sur une autre structure en amont ou varier en fonction d'autres calculs en cours – on parle alors de "couplage de codes". Par exemple, pour simuler l'interface entre l'eau douce et l'eau salée afin d'évaluer les risques d'infiltration du sel dans des ressources d'eau potable, un couplage est nécessaire entre deux phénomènes distincts : l'écoulement de fluide et le transport du sel, le premier conditionnant la diffusion du second et la concentration du sel influençant la viscosité du liquide [AYM99].

Les temps de calcul de la simulation sont variables mais dépendent de la complexité du problème à la fois en nombre de cellules du maillage, en itérations afin d'assurer la convergence des calculs et en pas de temps (si la simulation est temporelle). Les ressources matérielles mises à disposition des ingénieurs : ordinateur de bureau, grappe d'ordinateurs, supercalculateurs ; ont un fort impact sur la durée des calculs qui en règle générale durent plusieurs heures voire plusieurs jours. Afin d'ajouter un certain contrôle sur le bon déroulement de ces calculs, il est possible d'introduire une variable dont les fluctuations renseignent sur leur convergence. Des étapes de visualisation d'une telle variable au cours de la simulation peuvent ainsi être ajoutées dans cet optique sans pour autant mettre un terme aux calculs.

1.3 Visualisation

Les résultats issus de l'étape de simulation doivent être enfin analysés afin de comprendre et de valider le phénomène étudié. Ces résultats pouvant s'exprimer sous diverses formes : scalaire, vecteurs, matrices,... ; la visualisation scientifique propose un ensemble de méthodes génératrices de représentation bi-, tri- voire quadridimensionnelles permettant visuellement d'expliciter de tels résultats. La visualisation est donc un domaine vaste regroupant quasiment une méthode différente de représentation pour chaque type de données et domaine applicatif afin de répondre au mieux aux attentes d'un public expert. Nous évoquerons un ensemble non exhaustif de telles techniques dans la section 3.3 spécialisées entre autres pour les maillages tétraédriques.

Néanmoins, la création de telles images permet ainsi à l'ingénieur dans un premier de temps d'exploiter sa simulation, c'est-à-dire de détecter les possibles erreurs (nombre d'itérations de convergence insuffisant, résultats aberrants,...). Si de telles erreurs sont remarquées alors le processus d'élaboration du maillage ou les calculs sont modifiées et une nouvelle visualisation sera nécessaire après la prise en considération des modifications. Si de telles erreurs ne sont pas détectées alors ces images permettent dans un premier temps de comprendre la simulation puis dans un second temps de diffuser à plus grande échelle les résultats afin de valider les objectifs atteints et soulever de nouvelles problématiques. Ces deux étapes cruciales de la visualisation sont détaillées au sein de la section 3.1.

La visualisation intervient ainsi tout au long de l'élaboration d'une étude, permettant à la fois la validation visuelle de la discrétisation de l'espace, de la convergence des simulations (si cela est possible) et enfin des résultats obtenus. Il n'en reste pas moins important de remarquer que la visualisation scientifique obtient son rôle majeur en fin de processus pour l'exploitation et la diffusion des résultats.

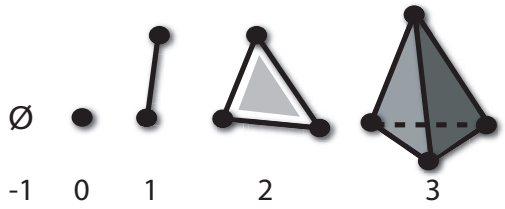


FIG. 1.6: **Simplexes dans $\mathbb{R}^3$**. De gauche à droite : l'ensemble vide, le sommet, l'arête, le triangle et le tétraèdre.

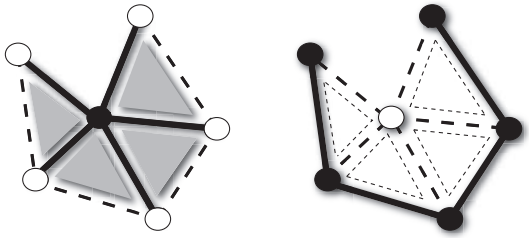


FIG. 1.7: **Lien et Étoilé d'un sommet dans un maillage triangulé**. À gauche : Les arêtes noires et les triangles gris appartiennent à l'étoilé du sommet central. À droite : Les arêtes et les sommets noirs appartiennent au lien du sommet central.

2 Maillages Tétraédriques : Notions et Définitions

Afin de réaliser une simulation numérique de phénomènes dans le cadre d'une étude scientifique, l'espace doit être dans un premier temps discrétisé. Nous avons vu que de nombreuses solutions sont possibles au sein de la section 1.1. Dans le cadre de cette thèse, nous nous intéressons principalement à un type de maillages irréguliers utilisés pour le calcul de simulations : les maillages tétraédriques. Ces maillages présentent en effet certains avantages comme leur génération rapide mais aussi des propriétés géométriques intéressantes que nous détaillons dans la suite de cette section.

Nous détaillons dans un premier temps la notion de complexe simplicial (section 2.1) puis celle de qualité d'un maillage tétraédrique (section 2.2) avant de citer quelques structures de données adaptées au stockage des maillages tétraédriques (section 2.3).

2.1 Complexe Simplicial

Bien que les définitions suivantes sont valables dans $\mathbb{R}^n$, n quelconque, nous nous restreignons sans perte de généralité à $n \leq 3$ puisque nous travaillerons dans cet espace par la suite.

Définition 1.1 Simplexe : On définit un simplexe σ comme l'enveloppe convexe de points affinement indépendants. Si d est le nombre de points définissant σ alors on dit que σ est un d-simplexe, d est appelée sa dimension et on note $dim\ \sigma = d$. On dit que σ est engendré par les points.

Dans $\mathbb{R}^3$, les *simplexes* possibles sont illustrés dans la figure 1.6 : l'ensemble vide, de dimension -1, le sommet de dimension 0, l'arête de dimension 1, le triangle de dimension 2 et le tetraèdre de dimension 3.

De plus, on appelle *cellule* le simplexe de dimension maximale. Lorsque $n = 2$, une *cellule* est un triangle. Lorsque $n = 3$, une *cellule* est un tétraèdre. Si σ est un simplexe généré par d points indépendants, alors tout simplexe τ généré par un sous-ensemble de l points est appelé une l-face et on note $\tau \leq \sigma$. Par abus de langage, on appelle une 2-face en dimension 3 une *face*, les 0-faces et les 1-faces sont respectivement les sommets et les arêtes.

Définition 1.2 Complexe Simplicial : Un complexe simplicial est une collection finie Σ de simplexes telle que :

(i) $\sigma \in \Sigma$ et $\tau \leq \sigma \Rightarrow \tau \in \Sigma$.

(ii) $\sigma, \nu \in \Sigma \Rightarrow \sigma \cap \nu \leq \sigma$ et $\sigma \cap \nu \leq \nu$.

Définition 1.3 Étoilé : L'étoilé d'un simplexe σ est l'ensemble de tous les simplexes qui contiennent σ :

$$Etoile\ \sigma = \{\tau \in \Sigma | \sigma \leq \tau\}$$

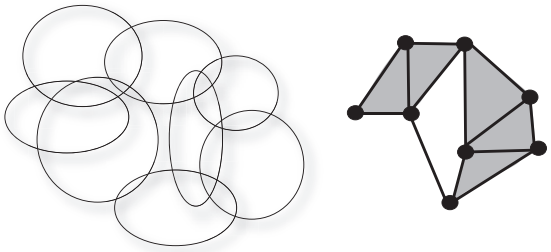


FIG. 1.8: **Nerf d'un ensemble quelconque.** Soit C un ensemble composé de 8 sous-ensembles $C_1,\ldots,C_8$ (à gauche). Le *nerf* de $C_1,\ldots,C_8$ correspond à un complexe simplicial dont les sommets correspondent aux sous-ensembles et chaque simplexe de plus haute dimension aux intersections non vides entre ces sous-ensembles. Ici, les intersections concernent deux voire trois sous-ensembles. Le nerf résultant sera composé d'arêtes et de triangles (à droite).

Définition 1.4 Lien : Le lien d'un simplexe σ est l'ensemble des l-faces de l'étoilé de σ qui n'intersectent pas σ :

$$Lien\ \sigma = \{\tau \in \Sigma / \exists \nu \in Etoile\ \sigma, \tau \leq \nu \text{ et } \tau \cap \sigma = \emptyset\}$$

On définit enfin la notion de *nerf* illustrée en figure 1.8.

Définition 1.5 Nerf : Soit C un ensemble et $C_1,\ldots,C_m$ m sous-ensembles de C. On définit le nerf de $C_1,\ldots,C_m$ et on le note $Nerf(C_1,\ldots,C_m)$ comme le complexe simplicial dont les sommets correspondent bijectivement aux parties $C_1,\ldots,C_m$ et composé des k-simplexes engendrés par $C_{i_1},\ldots,C_{i_k}$ tels que $\bigcap_{j=1}^{k} C_{i_j} \neq \emptyset$.

Le lecteur intéressé peut retrouver ces définitions et d'autres concepts associés aux complexes simpliciaux dans [Ede01].

2.2 Critère de qualité

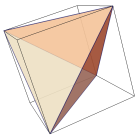
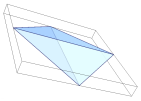
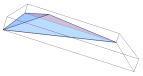
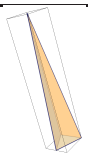
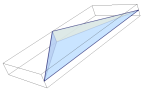
La qualité des tétraèdres au sein d'un maillage est primordiale afin de pouvoir appliquer des méthodes numériques permettant de simuler des phénomènes. En effet, des tétraèdres dégénérés peuvent conduire à une divergence des calculs. Un *bon* maillage de l'espace sera donc un maillage ne contenant pas ces tétraèdres dégénérés. Afin de déterminer de manière précise la présence de tels éléments dans un maillage tétraédrique, des mesures de qualité doivent être définies. Néanmoins, cela reste un post-traitement à l'obtention d'un maillage tétraédrique *via* un nuage de points ou une surface. De nombreux algorithmes ont été élaborés dans le but de construire un maillage garantissant de bonnes mesures de qualité [She98, YS00, TMFR05]. Nous détaillons une méthode possible basée sur les maillages dit de *Delaunay*.

2.2.1 Mesures de qualité

Afin de pouvoir caractériser les tétraèdres dégénérés, des mesures de qualité sont nécessaires. Elles permettent ainsi une classification de ceux-ci. [BE95] se base sur les angles diédral et solide alors que [CDE+00] regarde la dégénérescence au niveau des triangles pour réaliser sa classification. Par la suite, nous ne détaillons que quelques mesures de qualité référencées dans la table 1.1 qui permettent de détecter un certain nombre de ces dégénérescences comme les *slivers*, *cap*, *needle* et *wedge*. (cf. table 1.1).

Définition 1.6 Tétraèdre Sliver (Éclat) : Un *sliver* est un tétraèdre dont les quatre sommets sont géométriquement proches d'un plan et dont la projection orthogonale sur ce plan est un quadrilatère sans arête courte.

Les *slivers* sont les tétraèdres dégénérés les plus connus car la création de maillages satisfaisant le critère de *Delaunay* introduit de tels tétraèdres durant la construction. Une phase dédiée est donc généralement réalisée après la création du maillage afin d'éliminer ces cellules. Plus de détails sont donnés dans la sous-section suivante.

	Equilatéral	*Sliver*	*Cap*	*Needle*	*Wedge*
					
Angle Diédral min	70,5	15,9	15,8	53,3	11,4
Angle Diédral max	70,5	157,4	149,6	87,4	90
Angle Solide min	31,5	9,26	2,87	1,13	3,35
Angle Solide max	31,5	9,26	260,1	55,6	81,9
Ratio min/max long. arête	1	0,71	0,26	0,19	0,1
Ration Rayon-Arête	1	1,15	1,97	2,28	8,75

TAB. 1.1: **Tétraèdres de différentes qualités :** De gauche à droite : le tétraèdre *équilatéral*, le *sliver*, le *cap*, le *needle* et le *wedge*. À chaque tétraèdre est associé différents critères de qualité. L'*équilatéral* étant le tétraèdre parfait et les quatre autres les plus dégénérés.

Définition 1.7 Angle Diédral (ou Dièdre) : C'est l'angle formé par l'intersection de deux faces à arête commune du tétraèdre. Il y a six angles diédrals différents au sein d'un tétraèdre.

Définition 1.8 Angle Solide d'un tétraèdre : Si c est un sommet du tétraèdre et f sa face opposée alors l'angle solide est le rapport entre l'aire de projection centrale de f sur la sphère de centre c et le carré du rayon de la sphère.

Définition 1.9 Ratio d'Aspect : Il est défini comme le rapport entre la longueur de la plus grande arête et le rayon de la sphère inscrite. Il est souvent normalisé de manière à ce qu'il soit égal à 1 pour les tétraèdres équilatéraux.

Définition 1.10 Ratio Rayon-Arête : Il est défini comme le rapport entre le rayon de la sphère circonscrite et la longueur de la plus petite arête.

De nombreuses autres mesures de qualité peuvent être calculées afin de caractériser un tétraèdre : ratio entre la plus grande et la plus petite arête, aspect bêta, aspect gamma ou aspect de Frobenius [SEK+07].

Néanmoins, parcourir le maillage afin de localiser de tels tétraèdres puis les traiter afin de les enlever reste un processus fastidieux. Il est préférable d'optimiser les mesures de qualité lors de la construction du maillage. De nombreux algorithmes ont été proposés pour répondre à cet objectif. Nous détaillons l'un d'entre eux ci-après.

2.2.2 Maillage de Delaunay

Créer un maillage de *Delaunay* permet de minimiser la création de tétraèdres dégénérés grâce aux contraintes que le critère de *Delaunay* impose. C'est pour cela qu'elle est une des techniques de création de maillage les plus utilisées.

Définition 1.11 Simplexe de Delaunay : Un d-simplexe σ est dit de Delaunay si la d-sphère circonscrite à σ ne contient aucun autre sommet $v \notin \sigma$.

Définition 1.12 Tétraédralisation de Delaunay : Une tétraédralisation $\mathcal{D}$ est dite de Delaunay si elle est un complexe simplicial Σ composé de simplexes de Delaunay et que l'ensemble de ces simplexes recouvre l'enveloppe convexe des sommets. Un simplexe σ d'un sous-complexe $\mathcal{T}$ de Σ est dit localement de Delaunay si celui-ci est de Delaunay dans $\mathcal{T}$.

Définition 1.13 Opération de basculement (Flip) : Le basculement est une opération qui transforme un ensemble de simplexes qui n'est pas localement de Delaunay en un autre ensemble de simplexes qui est localement de Delaunay. La figure 1.9 illustre deux des différents basculements réalisables en trois dimensions.

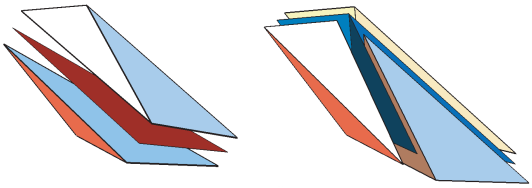


FIG. 1.9: **Basculement d'arêtes**. A gauche : deux tétraèdres sont unis par une face rouge *non localement de Delaunay*. On applique un basculement d'arêtes ($2 \rightarrow 3$) qui transforme les deux tétraèdres en trois nouveaux tétraèdres. A droite : Le résultat du basculement. Les trois faces bleues créées sont *localement de Delaunay*. La transformation inverse est possible (basculement dit $3 \rightarrow 2$).

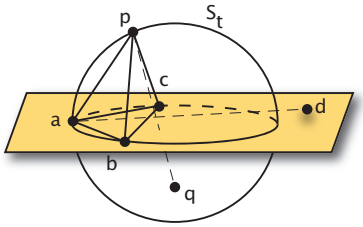


FIG. 1.10: **Tétraèdre contraint de Delaunay**. Le polygone orangé représente un polygone appartenant à $\delta\Sigma$, la surface à laquelle on veut contraindre notre maillage de Delaunay. p et q ne sont pas visibles entre eux. c et d par contre le sont. S_t est la sphère circonscrite de $t = (a,b,c,p)$ qui contient q mais pas d. t est constraint de Delaunay.

Il a été prouvé que les tétraédralisations de Delaunay peuvent être construites par l'insertion de points et l'utilisation d'opérations de basculement [Joe91].

Propriété 1.1 Borne Supérieure : *Les tétraédralisations de Delaunay garantissent que le ratio rayon-arête (définition 1.10) possède une borne supérieure dépendant uniquement des sommets du maillage. Cela permet de garantir que certains tétraèdres dégénérés ne peuvent être produits lors de la tétraédralisation.*

Malheureusement, les *slivers* ne sont pas écartés par cette borne supérieure car ce critère ne permet pas de les différencier des tétraèdres non dégénérés. Un post-traitement est donc nécessaire afin de les enlever. Un raffinement de Delaunay [She98], par exemple, divise ces tétraèdres problématiques de façon à les faire disparaître.

Définition 1.14 Diagramme de Voronoï : On définit le diagramme de Voronoï $\mathcal{V}$ d'un ensemble de points $p \in P$ dans l'espace E comme l'ensemble des cellules $V(p)$ telle que :

$$\mathcal{V}(P) = \bigcup_{p \in P} V_p = \bigcup_{p \in P} \{x \in E / \forall q \in P \; d(x,p) \leq d(x,q)\}$$

Le nerf du diagramme de Voronoï $\mathcal{V}$ (plus communément appelé le dual) n'est autre que la triangulation de Delaunay $\mathcal{D}$ basée sur les points $p \in P$.

$$Nerf(\mathcal{V}(P)) = \mathcal{D}(P)$$

Néanmoins, les tétraédralisations de Delaunay ne sont utiles que lorsque les données de bases sont des nuages de points. Pour certaines simulations, en mécanique par exemple, les maillages tétraédriques doivent être construits à partir de modèle CAO, donc de lignes ou de surfaces existantes – comme évoqué au sein de la section 1.1. Ainsi, le bord des maillages tétraédriques est contraint à une surface de bord $\delta\Sigma$. Afin de répondre à cette problématique, des maillages dit *contraints de Delaunay* peuvent être générés.

Définition 1.15 Visibilité entre deux sommets : La visibilité entre deux sommets p et q de Σ, est **occultée** s'il existe un polygone contraint $f \in \delta\Sigma$ tel que p et q se trouvent chacun d'un côté du plan et que le segment $[p,q]$ intersecte f. Sinon on dit que p et q sont visibles l'un par rapport à l'autre.

Définition 1.16 Tétraèdre contraint de Delaunay : Un tétraèdre t de Σ est dit contraint de Delaunay si sa sphère circonscrite ne contient aucun sommet visible autre que les sommets composant t. Cela veut dire que la sphère circonscrite peut contenir des points occultés. La figure 1.10 illustre cette définition.

Définition 1.17 Tétraédralisation Contrainte de Delaunay (CDT) : Une CDT est une tétraédralisation de Delaunay dont tous les tétraèdres sont contraints de Delaunay et qui est de plus contrainte à une surface, c'est à dire que les simplexes de la surface de bord sont imposés lors de la construction.

Une CDT ne vérifie qu'une partie des propriétés des tétraédralisations de Delaunay à cause des contraintes supplémentaires imposées en surface. Néanmoins, elle permet d'utiliser encore le raffinement de Delaunay et garantit une borne inférieure sur la longueur des arêtes générées [She02].

2.3 Structures de Données

Afin de représenter en mémoire les maillages tétraédriques obtenus, des structures de données adaptées sont nécessaires. Nous détaillons ci-après les plus utilisées. Le lecteur intéressé peut se référer à [DFH05] pour plus d'informations.

Soupe de cellules La *soupe de cellules* est la structure la plus générale possible. Elle permet de représenter chaque cellule de manière indépendante. Elle est surtout utilisée pour des traitements dit en mémoire externe lorsque le maillage ne peut pas être stocké entièrement au sein de la mémoire centrale de l'ordinateur – cf. section 5.1.1. Elle consiste à représenter un tétraèdre (et plus généralement tout polygone, polyèdre,...) par la position géométrique de ses quatre (n) sommets. La soupe de polygones représente donc un maillage tétraédrique par une liste de 4-tuples de vecteurs tridimensionnels.

La généralité (point fort) de cette structure de données en fait une grande consommatrice de mémoire (point faible).

Structure indexée centrée cellule La *structure indexée* centrée cellule est adaptée pour la représentation de complexes simpliciaux 3-variétés et permet de minimiser la place mémoire utilisée par rapport à la soupe de polygones. Elle consiste à stocker au sein d'un tableau indexé de 0 à $|V|-1$ la position géométrique des sommets. Les tétraèdres sont alors représentés au sein d'une liste où chaque sommet est référencé par un indice du tableau. La liste est donc composée de 4-tuples d'entiers.

Cette structure est étendue en ajoutant un certain nombre de relations d'adjacence lorsque la topologie du maillage est nécessaire. On parle ainsi de *structure indexée avec adjacence* lorsque celle-ci stocke en plus les voisins (tétraèdres partageant une face) de chaque tétraèdre sous la forme d'un 4-tuple de pointeurs. D'autres relations d'adjacences peuvent être ajoutées si nécessaire : sommet vers un tétraèdre, voisins d'arêtes, etc...

Structures topologiques Le stockage des relations d'adjacence entre les différents simplexes d'un maillage – on parle aussi de topologie ; provoque un surcoût mémoire non négligeable. Un certain nombre de structures ont ainsi été développées afin d'obtenir un compromis entre efficacité et occupation de la mémoire centrale. Elles sont principalement des généralisations des structures par demi-arêtes (DCEL) [MP78] pour les maillages triangulaires. Par la suite, on pose $t = |T|$ le nombre de tétraèdres composant le complexe simplicial Σ. [LT97] propose une structure générique : la *Handle-Face* de complexité $O(135\,t)$. [LLLV05] spécialise sa structure : la *Compact Half-Face (CHF)*, aux complexes simpliciaux 3-variétés afin d'obtenir une complexité réduite à $O(8\,t)$.

3 Maillages Tétraédriques et Visualisation

Une fois la discrétisation de l'espace obtenue grâce à la génération du maillage tétraédrique, la simulation numérique est réalisée. Les résultats générés après quelques heures (voire jours) de calcul sont alors visualisés. Nous détaillons au sein de cette section les notions fondamentales associées à cette étape-clef qu'est la visualisation en terme d'objectifs, d'efficacité et de techniques afin de répondre pleinement aux attentes des utilisateurs.

Nous précisons tout d'abord les deux étapes distinctes composant les objectifs de la visualisation (section 3.1), puis la notion d'interactivité (section 3.2) avant de détailler un panel non exhaustif de techniques de visualisation (section 3.3).

3D Interactive	30 images par seconde
temps-réel	$\geq$ 5 images par seconde
quasi-temps-réel	$\geq$ 1 image par seconde
interactif	$\geq$ 0,1 image par seconde

TAB. 1.2: **Temps-réel et interactivité**. Les termes temps-réel et interactif sont utilisés de manière informel en informatique graphique. Une classification est tout de même possible en fontion du taux de rafraîchissement des applications [Sai94].

3.1 Exploitation et Diffusion

La visualisation scientifique, comme nous l'avons déjà précisé en section 1.3, intervient principalement en fin de simulation numérique afin d'aider l'ingénieur à exploiter puis diffuser les résultats obtenus sur le maillage tétraédrique. Ces tâches complémentaires sont en réalité très différentes et représentent deux étapes distinctes importantes qui requièrent des besoins spécifiques. La première est dite *étape d'exploitation* alors que la seconde sera plutôt dite *de décision*.

L'exploitation est faite juste après la fin de la simulation, au moment où l'ingénieur récupère les résultats de ses calculs. Il ne sait donc pas a priori ce qu'il va découvrir même si son expertise lui permet de faire un certaine nombre d'assertions qui pourront l'aider. Cette première étape doit donc permettre à la fois l'extraction des tendances générales et le repérage rapide des zones d'intérêts (comme les points critiques) avec les différentes raisons qui pourraient en être la cause. Pour cela, l'exploitation est réalisée grâce à une phase d'exploration. Cette exploration, correspondant à une manipulation des données (translation, rotation, zoom, extraction de régions, ...), doit être fluide grâce à la génération d'images en temps-réel. Cette étape est cruciale car elle va permettre à l'ingénieur expert de valider ou de réfuter son modèle, de localiser les zones où la simulation est incorrecte ou au contraire ayant des comportements inattendus ou dignes d'intérêt.

Lorsque cette étape se conclut par une première validation, il est nécessaire de présenter les résultats soit à d'autres ingénieurs spécialistes, soit à un public moins averti – si ce n'est le grand public ; afin de confronter les points de vue et prendre les décisions qui s'imposent suite aux résultats. Comme la phase d'exploitation est terminée, les zones d'intérêt sont maintenant connues et l'ingénieur peut alors se concentrer sur la mise en valeur et la qualité des images générées afin de communiquer les résultats le plus clairement possible. Cette étape ne nécessite pas de temps-réel (cf. section 3.2) à proprement parler car l'utilisateur peut se permettre d'attendre plusieurs secondes avant la génération d'une image.

Dans le cadre de cette thèse, nous nous intéressons principalement à la première étape de visualisation qui s'inscrit pleinement au sein de la boucle de la découverte scientifique : l'exploitation des données *via* l'exploration interactive des maillages tétraédriques.

3.2 Temps Réel et Interactivité

Le *temps-réel* est défini comme un "mode d'utilisation d'un ordinateur dans lequel le système doit fournir une réponse dans un délai fixé en fonction de l'application : d'une fraction de seconde (pilotage d'un engin) à plusieurs heures (météorologie)."[4]

Néanmoins, contrairement aux domaines de l'interface homme-machine ou des systèmes embarqués, en informatique graphique, les termes de *temps-réel* et d'*interactivité* sont utilisés de manière informel. Par abus de langage, on définira une application interactive comme une application temps-réel (et réciproquement).

En visualisation scientifique [Sai94] propose une définition précise de ces termes en 1994. Elle se base uniquement sur le taux de rafraîchissement des applications.[5] Les distinctions suivantes regroupées au sein du tableau 1.2 peuvent ainsi être considérées.

Un taux de rafraîchissement d'au moins 30 images à la seconde sera considéré comme de la "*3D Interactive*". Un tel taux garantit l'intervention instantanée d'un utilisateur au sein de l'application. Son maintien est

[4]Définition issue de *Del-Perret 1973*, disponible à http ://www.cnrtl.fr/definition/temps.

[5]ou plus communément d'images produites à la seconde, en anglais *frame per second* (fps).

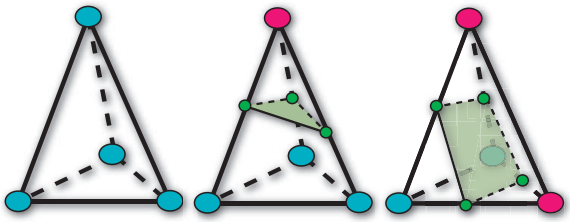


FIG. 1.11: **Marching Tetrahedra.** De gauche à droite : les trois configurations pour l'extraction d'une surface au sein d'un tétraèdre. La couleur des sommets représentent les polarités par rapport à l'isovaleur. À gauche : l'isosurface n'intersecte pas le tétraèdre. Au milieu : extraction d'un triangle. À droite : extraction de deux triangles.

essentiel au sein d'applications comme les jeux vidéos. On parlera de *temps-réel* si le temps de rafraîchissement est supérieur à cinq images par seconde et de *quasi*-temps-réel si celui-ci dépasse une image à la seconde. En visualisation scientifique, ces taux de rafraîchissement sont essentiels lors de l'exploration des données afin de garantir une certaine fluidité de l'application. On parlera d'application *interactive* lorsque l'affichage prend au plus dix secondes. Cette attente semble raisonnable pour la production d'images destinées à la diffusion des résultats.

3.3 Les différentes techniques de visualisation

Pour pouvoir valider ou diffuser les résultats d'une simulation, la visualisation scientifique propose un ensemble de techniques permettant de créer des images porteuses de sens pour un public averti et pour les décideurs non spécialistes voire le grand public. Ces techniques de visualisation sont adaptées en fonction du type de données issu de la simulation : scalaire, vectoriel, tensoriel, ... Elles le sont aussi en fonction des objectifs de visualisation de l'utilisateur.

Nous présentons ci-après un bref aperçu de ces différentes techniques, regroupées par type de données. Nous définissons la notion de fonction de transfert (section 3.3.1) avant de préciser les techniques de visualisation pour les champs scalaires (section 3.3.2), vectoriels (section 3.3.3) et tensoriels (section 3.3.4).

3.3.1 Fonction de transfert

Une des premières étapes communes à de nombreuses techniques de visualisation est la transformation du champ scalaire réel en une information de couleur et de transparence. Pour cela, l'utilisateur définit une fonction de transfert ϕ qui transforme l'espace scalaire au sein de l'espace couleur (Rouge, Vert et Bleu) et transparence (Alpha) $\phi : \mathbb{R} \rightarrow < R,V,B,A >$[6]. L'élaboration de la fonction de transfert est une étape importante dans la compréhension puis la communication des résultats. Par exemple pour l'étude de température ou de pression, un dégradé de rouge (hautes valeurs) vers bleu (basses valeurs) semble adapté. En médical, la colorisation des organes par leur couleur naturelle est un exemple possible de choix.

Lorsque le champ d'attributs est vectoriel, la fonction de transfert ϕ peut être élaborée soit en couplant la technique de visualisation vectorielle avec un champ scalaire associé, soit par exemple en extrayant la norme de chacun des vecteurs. Pour des vitesses angulaires, la transformation de chacune des coordonnées en un canal de couleurs est aussi une autre solution.

3.3.2 Pour les champs scalaires

Les techniques de visualisation des champs scalaires ont été regroupées en deux grands groupes : les *méthodes indirectes* et les *méthodes directes* [RS01]. Les premières ne cherchent à représenter qu'un sous-ensemble des données initiales (plans de coupe, isosurface) alors que les secondes les traitent dans leur glo-

[6] L'espace $< R,V,B >$ peut aussi être réduit à un seul canal dit de *luminance* L afin de générer un niveau de gris.

balité (rendu volumique direct). L'ensemble de ces techniques est illustré au sein de la figure 1.12 pour la visualisation de la densité d'essence dans une chambre de combustion.

Plan de Coupe La visualisation par *plan de coupe* consiste à définir un plan au sein de la boîte englobante du volume par rapport auquel une coupe de l'ensemble de données sera réalisée. On distingue l'opération de *taille* (*clip*) qui consiste à afficher le volume entièrement derrière le plan de coupe à l'opération de *tranche* (*slice*) qui n'extrait que le plan de coupe avec l'affichage du champ scalaire sur ce plan via la fonction de transfert. Ces deux méthodes sont illustrées dans la figure 1.12 en haut à gauche.

Isosurface La visualisation par *surface isovaleur* (ou *isosurface*) consiste à extraire une surface équipotentielle du champ scalaire. Sa définition est la suivante :

Définition 1.18 Isosurface : la surface équipotentielle I_k pour une valeur scalaire k est définie par :

$$I_k = \{\mathbf{x} | F(\mathbf{x}) = k\}$$

où F est un champ scalaire.

L'extraction d'une isosurface à partir d'un maillage tétraédrique est réalisée via l'application d'un *Marching Tetrahedra* [GH95], version tétraédrique du *Marching Cube* [LC87] qui réalise le calcul des surfaces équipotentielles pour les grilles régulières. Trois cas différents sont possibles et illustrés au sein de la figure 1.11. Contrairement au *Marching Cube*, il n'y a pas de cas ambigu d'extraction.

La classification de ces situations d'extraction et l'indépendance du traitement par cellules a permis la transposition de l'algorithme du *Marching Tetrahedra* au sein des différentes générations des cartes graphiques programmables avec des gains significatifs [Pas04, BCL06, TSC07].

L'extraction de surfaces équipotentielles a été réalisée au sein de la figure 1.12 en bas à gauche. Plusieurs potentiels ont été calculés puis modulés via l'utilisation de transparence afin de limiter les occlusions, limitation principale d'une telle méthode. En effet, l'extraction de plusieurs isosurfaces peut entraîner des occlusions qui peuvent limiter la compréhension de la simulation. Une solution à ce problème est d'utiliser à la place une méthode directe comme le rendu volumique direct.

Rendu Volumique Direct Le *rendu volumique direct* [KVH84] (ou DVR pour *Direct Volume Rendering*) est une méthode directe simulant l'intégration de la lumière traversant le volume de données assimilé à un milieu translucide combinant un effet d'éclairage et un effet d'opacité. Cette intégration est basée sur la fonction de transfert ϕ. La modulation selon la transparence permet ainsi de visualiser l'ensemble de données dans son entier tout en intégrant les occlusions entre les différentes parties de celui-ci le long des rayons lumineux.

Des alternatives existent qui sont utilisées principalement dans le domaine médical. La *méthode de projection de l'intensité maximale* [HMS95] (ou MIP pour *Maximum Intensity Projection*) conserve uniquement l'intensité maximale le long du rayon. La méthode des *rayons X* [DCH88], quant à elle, n'applique aucune atténuation le long du rayon, c'est-à-dire qu'elle ne prend pas en compte la transparence.

Dans la suite de cette thèse, nous nous intéressons uniquement qu'au rendu volumique direct ordonné. Une telle technique est illustrée au sein de la figure 1.12 à droite et celui-ci sera détaillé plus en profondeur au sein du chapitre 4.

3.3.3 Pour les champs vectoriels

La visualisation de champs vectoriels a été regroupée en trois grands groupes : les *méthodes directes*, les *méthodes denses texturées* [LHD+04] et les *méthodes géométriques* [PVH+02]. Les figures 1.13 et 1.14 représentent un échantillon de telles techniques.

Méthodes directes Les méthodes directes représentent le champ vectoriel en minimisant les précalculs. Les représentations par flèches (en hérisson ou *hedgehog*) ou *via* des glyphes plus complexes, avec une colorisation par la norme du champ, permettent ainsi une exploration immédiate de la simulation de flux.

Une représentation grâce à des flèches orientées selon les directions du champ vectoriel et dont la taille est proportionnelle à la taille du vecteur local est présentée dans la figure 1.13 à gauche.

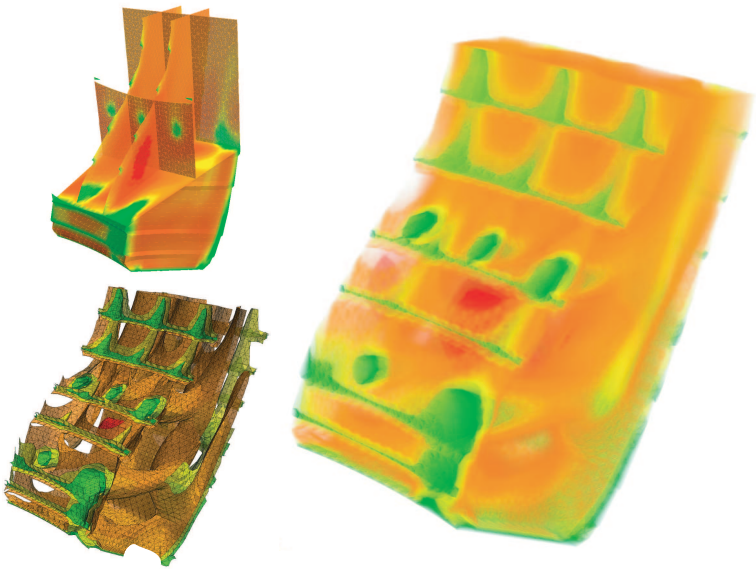


FIG. 1.12: **Techniques de visualisation pour un champ scalaire pour l'ensemble de données *Comb.*** Le champ scalaire représente la densité d'essence se trouvant au sein de cette chambre de combustion. Il est issu d'une simulation afin d'en vérifier le bon fonctionnement [SML06]. En haut à gauche : Plan de coupes (taille et tranches texturées). En bas à gauche : surfaces isovaleurs avec transparence. À droite : rendu volumique direct par lancer de rayons.

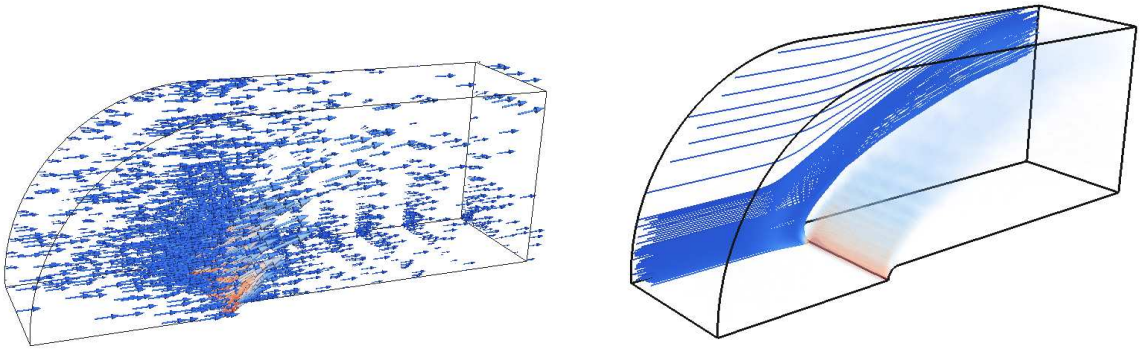


FIG. 1.13: **Techniques de visualisation pour un champ vectoriel pour l'ensemble de données *Blunt Fin.*** On étudie la simulation du trajet d'un courant d'air enfermé au sein d'une conduite. Le flux est censé être parallèle à la base [HB84]. À gauche : le champ vectoriel est représenté par un ensemble de flèches (*hedgehog*). Leur couleur est extraite d'un champ de densité associé, leur taille et leur inclinaison de la norme et de la direction du champ respectivement. À droite : le champ vectoriel est représenté par un ensemble de lignes de courant (*streamlines*) significatives. Elles sont immergées dans un rendu volumique direct afin d'aider à la compréhension du phénomène.

Méthodes denses texturées Ces techniques calculent une texture afin d'offrir une représentation dense du flux. Elles regroupent les techniques de *spot noise* qui consiste à déformer et répartir dans l'espace des ellipses (ou tout autre glyphe) ; la famille des techniques *LIC* (*Line Integral Convolution*) se basant sur une texture de bruit blanc convoluée à un noyau orienté selon la direction principale du champ : et l'advection de textures qui consiste à déplacer cette fois-ci les éléments des textures en fonction du champ vectoriel. La figure 1.14 illustre de telles approches sur la surface d'un avion.

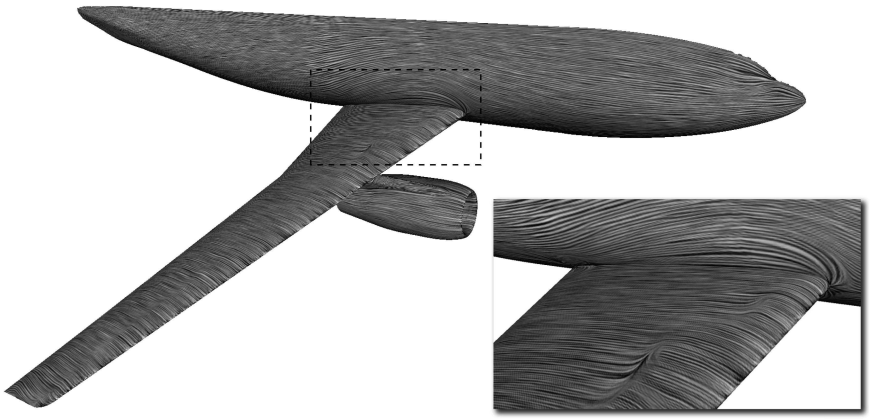


FIG. 1.14: **Méthodes denses texturées (*Line Integral Convolution*)** issue de [LTWH08]. Le champ vectoriel est représenté sur la surface de l'avion en utilisant la technique *Line Integral Convolution*.

Méthodes géométriques Les méthodes géométriques reposent sur une intégration préalable des données. La géométrie issue de ces intégrations reflète ainsi les propriétés du flux. Les *lignes de courant* : *streamlines*, *pathlines* ou *streaklines* en font parties, comme les *streamribbons* qui sont des streamlines plus épaisses permettant entre autres de mettre en évidence la vorticité d'un flux plus facilement.

Une représentation de streamlines extraites d'un champ vectoriel est proposée au sein de la figure 1.13 à droite. La colorisation des streamlines est issue d'un champ scalaire associé. Elles sont plongées au sein d'un rendu volumique de ce même champ scalaire.

3.3.4 Pour de plus hautes dimensionnalités

Lorsque les dimensionnalités des solutions des équations simulées sur le maillage dépassent les usuelles trois dimensions spatiales (comme l'obtention de tenseurs par exemple), la façon de représenter l'ensemble de ces dimensions devient plus complexe à mettre en œuvre. Néanmoins, de nombreuses solutions existent déjà pour répondre à cette demande.

L'utilisation de glyphes permet par exemple de représenter via des ellipses [LAK+98], des boîtes [JLZ+01] ou des *superquadrics* [Kin04] des données tensorielles en modifiant à la fois les paramètres les définissant, leur orientation et leur couleur par exemple.

Comme pour les champs vectoriels, des méthodes géométriques ont été aussi explorées comme les *hyperstreamlines* [DH93] qui sont des lignes de courant définies par les vecteurs propres du tenseur ou encore les *tensorlines* [WKL99] moins sensibles au bruit au sein des données.

Une dernière possibilité est d'utiliser la colorisation et l'illumination comme des représentations des tenseurs. Pour cela, la colorisation de l'espace est définie sur une sphère et l'éclairage est réalisé en fonction de la direction et de la courbure extraites du tenseur. Cette méthode a été appelée *hue-balls and lit-tensors* [KW99] dans le but de réaliser un rendu volumique direct de champs tensoriels de diffusion.

4 Carte Graphique : Notions et Définitions

L'arrivée récente au sein de la communauté de l'informatique graphique des nouvelles cartes graphiques a provoqué une révolution majeure dans la manière de résoudre une problématique et le monde de la visualisation scientifique (et plus récemment de la simulation numérique) a cherché à profiter pleinement de la puissance de calculs et de rendu d'un tel outil. La création d'une structure unifiée et la possibilité de pouvoir modifier la connectivité des maillages rend la carte graphique aujourd'hui incontournable dans le domaine de la visualisation scientifique tant sur le traitement des maillages en précalculs que sur l'interactivité des rendus afin d'en faciliter l'exploration et la compréhension.

Nous revenons donc dans cette section, sur les motivations (section 4.1) et les enjeux (section 4.3) liés à la carte graphique tout en présentant l'architecture dédiée de cet outil nommée le *pipeline graphique* (section 4.2).

4.1 Historique et Motivations

Devant l'augmentation croissante de la puissance de calculs des ordinateurs et leur accessibilité à la représentation graphique pour les jeux vidéos, la visualisation scientifique ou la réalité virtuelle, le temps-réel – voir section 3.2 ; est devenu une nécessité. Néanmoins, la conjecture de Moore concernant les processeurs révélait une limite à la puissance de calculs d'un seul ordinateur ne permettant pas d'atteindre le temps-réel en graphique. Le développement d'une unité spéciale pour le rendu graphique, la carte graphique ou *GPU (Graphics Processor Unit)*, est donc devenue un besoin afin de répondre aux attentes de ces utilisateurs.

La carte graphique est un regroupement matériel de plusieurs processeurs parallélisés et spécialisés afin de garantir les meilleurs performances pour la visualisation d'objets tridimensionnels modélisés par des surfaces complexes triangulées. Les différentes phases de traitement, du transfert de la géométrie à l'obtention d'une image, sont regroupées au sein du *pipeline graphique* – cf. figure 1.15. La redondance des opérations et surtout leur indépendance au sein du pipeline a justifié l'utilisation de processeurs travaillant en parallèle afin d'assurer un meilleur rendement que le processeur principal (CPU) de l'ordinateur. Les processeurs graphiques ont de plus été dédiés en fonction des différentes étapes du pipeline. Ainsi, les fragments habituellement plus nombreux que les primitives géométriques se sont vus assigner plus de processeurs pour améliorer les performances. Fort de son succès, l'architecture des cartes graphiques a connu de nombreuses mutations au cours des dernières années provoquées par les besoins spécifiques des industriels.

Le GPU est devenu au cours des années 2000 une unité de calcul efficiente capable de réaliser plusieurs GigaFLOPS[7] – et même maintenant quelques TéraFLOPS –, concurrençant sans équivoque les ordinateurs de bureau monocœur les plus puissants et même certains multicœurs. Par exemple, fin 2008, les ordinateurs de bureau quadricœurs atteignaient les 70 GigaFLOPS alors que les GPU réalisaient déjà $1,2$ TéraFLOPS. Une telle puissance de calcul a naturellement interpellé des spécialistes novices en graphique, résultant à la génération de nouvelles cartes graphiques dites *unifiées*, c'est-à-dire architecturalement indépendantes du pipeline graphique – bien que celui-ci reste toujours implantable. L'utilisation dans d'autres domaines comme la simulation et la parallélisme de ces processeurs est ainsi favorisée, renforcée récemment par la livraison de processeurs graphiques à double précision pour les nombres flottants, permettant ainsi d'égaler la précision des processeurs des ordinateurs.

4.2 Le Pipeline Graphique

Le pipeline graphique est représenté au sein de la figure 1.15. Il traite en entrée la géométrie sous la forme de sommets et de primitives. Des informations, appelées *attributs*, peuvent être ajoutées pour chaque sommet comme une couleur, des coordonnées de texture, une normale, etc... En sortie, une image composée de pixels est affichée à l'écran.

Au sein de ce pipeline, les sommets sont transformés lors d'une première étape. Leurs coordonnées tridimensionnelles dans l'espace objet sont modifiées en des coordonnées tridimensionnelles dans l'espace image. L'illumination via l'utilisation des normales est aussi calculée. Les primitives sont ensuite formées dans une deuxième étape. Les attributs associés à chaque sommet sont interpolés au sein des primitives avant que celles-ci ne soient envoyées à l'étape de tramage qui va transformer la géométrie en *fragments*. Les *fragments* sont des *pixels* ayant une coordonnée de profondeur nécessaire pour la mise-à-jour du tampon de profondeur (*z-buffer*) et la gestion de l'occlusion. Les fragments sont ensuite colorés, ordonnés et transformés en pixels pour créer l'image finale. En fonction du rendu, les opérations sur les fragments peuvent être diverses : plaquage de textures, prise en compte de la transparence, ...

Ce pipeline théorique a été implanté sur les cartes graphiques et a connu un certain nombre de mutations le conduisant à sa forme actuelle illustrée par la figure 1.16. De nombreuses améliorations ont été apportées afin d'accélérer la communication entre le CPU et le GPU. De plus, certaines étapes sont devenues directement programmables afin de diversifier les techniques de transformations géométriques et de rendu. La communication avec la carte graphique est réalisée via des interfaces de programmation[8] comme *OpenGL* [SWND07] ou

[7] FLOPS : *FLoating-point Operations Per Second* ou opérations à virgule flottante par seconde est une mesure de la vitesse des microprocesseurs.

[8] ou plus communément API pour *Application Programming Interface*.

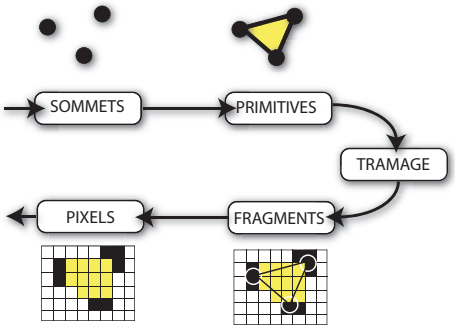


FIG. 1.15: **Pipeline graphique.** Le pipeline graphique est composé essentiellement de cinq phases. La première considère les sommets individuellement, la deuxième construit les primitives géométriques sous forme de lignes ou de triangles, la troisième est le tramage (ou *rasterisation*) convertissant la géométrie en fragments (pixels tridimensionnels), la quatrième effectue les opérations de colorisation, de textures et de profondeur au niveau des fragments, enfin la cinquième affiche les pixels à l'écran.

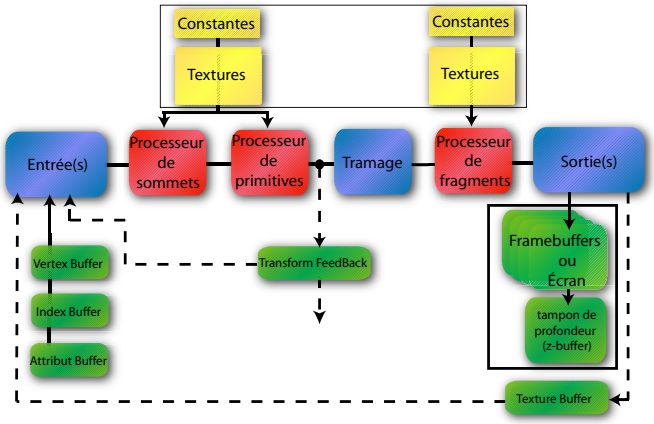


FIG. 1.16: **Pipeline programmable des cartes graphiques.** Actuellement, les étapes de traitement des sommets, des primitives et des fragments sont modifiables (en rouge) grâce à l'écriture de *shaders*. La phase de tramage (en bleu), quant à elle, est implantée de manière fixe. Les pixels et la géométrie sont récupérables au sein de différents tampons (*buffers* en vert). La carte graphique permet de plus de stocker un certain nombre d'informations au sein de tampons et de textures (en vert et jaune).

DirectX [Mic09a].

Afin d'envoyer l'information géométrique de manière optimisée à la carte graphique, des tampons (*buffers* en vert) sont utilisés contenant la géométrie (*vertex buffer objects ou VBO*), la connectivité sous forme d'indices (*index buffer*) et une multitude d'attributs (*attribut buffer*).

D'autres informations peuvent aussi être envoyées à la carte graphique comme des constantes et des textures (en jaune). Une texture est une image uni-, bi- ou tridimensionnelle composée de *texels*[9]. Elles sont souvent utilisées comme tableau afin de stocker des informations pouvant ne pas être visuelles – comme, par exemple, de la géométrie. Ces mêmes textures sont utilisées pour stocker l'image finale issue du pipeline dans le cadre de rendu *off-screen* au sein de la mémoire écran (*framebuffer objects ou FBO*) et du masque de profondeur. Pour l'envoi et le rapatriement de ces textures, certains tampons ont été spécialisés (*pixel buffer objects ou PBO*).

Une fois ces informations stockées au sein de la carte, celles-ci vont subir les transformations décrites précédemment. Mais, pour le traitement des sommets, des primitives et des fragments, des processeurs configurables ont été introduits (en rouge). Le programmeur peut ainsi spécifier les différentes actions réalisées au sein de chacun des processeurs.

Le processeur de sommets (*Vertex Shader Unit*) et le processeur de primitives (*Geometry Shader Unit*) s'occupent de la géométrie : des primitives simples comme des points, des lignes ou des triangles bien que les *API* permettent l'envoi de primitives géométriques plus complexes. Ils permettent entre autres de déplacer les sommets ou de modifier les primitives géométriques – par exemple des triangles en lignes. Ces modifications peuvent être récupérées sans passer par la phase de tramage en utilisant un tampon spécialisé (*transform feedback buffer*).

Si le tramage n'est pas désactivé, les fragments sont traités au sein du processeur de fragments (*Fragment Shader Unit*) qui de plus définit les différents types d'écriture finaux : au sein de l'écran ou de textures. Là encore, ces informations peuvent être directement réutilisées en entrée d'un second pipeline graphique soit sous la forme de textures soit sous la forme d'un tampon grâce au tampon de texture (*texture buffer object ou*

[9]Un *texel* (*Texture Element*) est un élément de la texture par analogie au pixel qui est un élément d'une image (*Picture Element*) et au voxel qui est un élément d'un volume (*Volume Element*). Un texel contient de un à quatre canaux, traditionnellement les trois canaux de couleurs (**R**ouge, **V**ert et **B**leu) et un canal de transparence (**A**lpha). Il peut être actuellement un entier (signé ou non) ou un réel sur huit, seize ou trente-deux bits par canal.

TBO).

L'architecture est ainsi pensée afin de limiter les transferts d'information bilatéraux entre le CPU et le GPU qui sont un goulot d'étranglement.

Des langages spécifiques ont été développés afin de permettre la création de programmes (*shaders*) exécutés au sein de chacun des processeurs spécifiques. Les plus utilisés sont *GLSL* (openGL Shading Language) [Kes09], Cg (C for Graphics) [FK03] et HLSL (High Level Shader Language) [Mic09b].

4.3 Unification Architecturale et Devenir

Depuis 2006, la carte graphique s'est unifiée architecturalement afin de permettre des calculs autres que ceux liés aux graphismes. Cela signifie que l'ensemble des processeurs (de sommets, de primitives et de fragments) ne sont plus spécialisés – cf. section 4.1. Cette unification fut initiée par un courant récent dit de "calcul générique sur processeur graphique" ou plus communément *GPGPU* (pour *General-Purpose on Graphics Processing Units*)[10].

Cette unification a ouvert de nouvelles opportunités à la fois pour le parallélisme et la simulation. Des langages haut niveau sont ainsi en train d'émerger comme *CUDA* (*Compute Unified Device Architecture*) [NVI07] ou *openCL* (*Open Computing Language*) [kowG09] ne se référant plus à des notions de graphisme. L'émergence de cartes sans unité de tramage et l'annonce de nouveaux CPU concurrentiels s'inspirant de leur architecture – le processeur *Larabee* de *Intel* – sont les premiers signes que les GPU semblent prendre une place de plus en plus importante au sein de l'informatique toute entière et ne se cantonnent plus au monde du graphique.

La carte graphique semble donc être l'unité de calcul incontournable au cours de ces prochaines années avec un potentiel de calcul dépassant sans équivoque les processeurs centraux de nos ordinateurs de bureau. Néanmoins, à l'heure actuelle, elle n'est pas non plus un *deus ex machina* permettant de résoudre tous les problèmes de complexité temporelle. La nécessité de transformer les algorithmes habituellement séquentiels en programme parallélisé peut entraîner une perte non négligeable des performances malgré la puissance de calcul. De plus, le rapatriement des données au sein du processeur central reste une demande coûteuse qui peut rapidement devenir le goulot d'étranglement d'une application. Il est donc prudent de signaler que bien que la carte graphique reste indéniablement une solution pour accélérer un algorithme, il est important de saisir les modifications et les goulots d'étranglements que pourrait entraîner une implantation sur carte graphique par rapport à une solution pleinement CPU.

5 Enjeux

La visualisation des maillages issus des simulations se confronte à plusieurs enjeux essentiels pour aider l'ingénieur ou le chercheur à analyser puis diffuser les résultats des calculs. Ces enjeux sont premièrement de garantir l'exploration interactive (cf. section 3.2) des données lors de leur visualisation et deuxièment d'aider l'utilisateur à analyser plus rapidement ses données en essayant d'incorporer des connaissances a priori.

Pour cela, les techniques de visualisation doivent prendre en compte autant la taille des données (section 5.1) que leur localisation géographique (section 5.2) que la localité de l'information d'intérêt (section 5.3).

5.1 La taille des données

La présence de supercalculateurs, de grappes d'ordinateurs dédiées à la simulation et récemment de grappes hybrides mélangeant cartes spécialisées (de type *Tesla*) et processeurs centraux[11] assure une puissance de calcul toujours plus massive afin de réaliser des simulations dont la précision est sans cesse améliorée. Cela résulte en la création de maillages et donc de résultats dont la taille croît chaque jour. Le tableau 1.3 indique quelques exemples de données massives obtenues récemment. Cette quantité d'informations atteindra sous peu

[10] http ://www.gpgpu.org

[11] avec la présence entre autres en France des supercalculateurs du centre *Jacques Louis Lions* regroupant les sites de l'*IDRIS* et du *CCRT* et du nouveau prototype hybride installé au *CINES*.

Domaine scientifique	Taille	Institut
Astronomie - Supernovae (TSI)	1 To	ORNL
Flux - Turbulences	3 To	LLNL
Astronomie - Projet *Mare Nostrum*	8 To	BSC
Combustion (Auto-Allumage)	35 To	ORNL

TAB. 1.3: **Exemples de simulations massives**. Les simulations actuelles atteignent des centaines de gigaoctets (10^9 octets), voire des dizaines de téraoctets (10^{12} octets). Demain, elles occuperont des pétaoctets (10^{15} octets), voire des exaoctets (10^{18} octets).

les pétaoctets, voire des exaoctets, ce qui représente de nouveaux défis.

En effet, afin de visualiser efficacement ces données massives, l'amélioration tant en espace mémoire qu'en complexité temporelle des techniques de visualisation est cruciale puisqu'elles sont fortement liées au nombre de primitives composant le maillage. L'utilisation de la carte graphique et/ou de grappes d'ordinateurs et/ou de méthodes dites *en mémoire externe* sont des solutions couramment utilisées de nos jours afin de répondre à cette problématique. Nous détaillons ci-après un ensemble non exhaustif de solutions proposées tant en mémoire externe (section 5.1.1) qu'avec une approche parallélisée (section 5.1.2).

5.1.1 Les approches en mémoire externe

Lorsque les données ne peuvent pas être chargées entièrement au sein de la mémoire vive de l'ordinateur, des solutions dites en mémoire externe (ou *out-of-core*) sont mises en place. Développées dans un premier temps pour les maillages surfaciques triangulés concernant la compression [HLK01, IG03], la simplification [Lin00, CMRS03] et la multirésolution dépendante du point de vue [DP02, YSGM04], la visualisation scientifique a aussi intégré un certain nombre d'approches en mémoire externe [SCESL02].

Les méthodes en mémoire externe s'inspirent principalement du papier fondateur de Clark [Cla76] sur le traitement des données ne pouvant être chargées entièrement en mémoire vive. Cox et Ellsworth [CE97] sont les premiers à les transposer pour la visualisation scientifique. Leur système repose sur une pagination construite sur le constat que pour de nombreuses techniques de visualisation (*streamlines*, plans de coupe, ...) seulement une sous-partie des données est nécessaire. Un tel système a aussi été proposé récemment pour le rendu volumique direct de grilles irrégulières [VCS+07].

Des structures de données sont adaptées afin d'accélérer un certain nombre de techniques : un *octree* [USM97] pour l'extraction de *streamlines*, des *R-tree* [LM98] (proposés pour contrecarrer le déséquilibre des *octrees* construits sur les grilles irrégulières) pour la projection de cellules, des schémas d'indexation hiérarchiques [PF01] pour des plans de coupe, ou des *interval trees* [CSS98] et des *Temporal Branch-On-Need octree* [SH00] pour l'extraction de surfaces isovaleur respectivement statiques ou temporelles. L'ensemble de ces structures permet de découper d'une manière adaptée à la technique l'espace des données de façon à accélérer la recherche et l'extraction des informations utiles.

Néanmoins, ces méthodes restent spécialisées à des techniques de visualisation particulières. À notre connaissance, aucune méthode en mémoire externe n'a été développée pour une approche multirésolution pour les grilles irrégulières tétraédriques qui sont le cadre de cette thèse.

5.1.2 Les approches parallèles

Une autre manière pour visualiser des ensembles de données massifs est de faire appel, lorsque c'est possible, à la puissance de calcul des grappes d'ordinateurs (et maintenant hybrides) et à leur mémoire distribuée. Pour cela, la technique de visualisation doit être découpée en sous-problèmes, chacun étant traité de manière indépendante par chaque ordinateur et/ou carte graphique de la grappe. L'élaboration de la stratégie afin de déterminer les sous-problèmes se résume à deux possibilités (en visualisation) décrites par Molnar *et al.* [MCDF94]. La première est un *tri précoce* (ou *sort-first* en anglais) sur les primitives géométriques en fonction de leur position spatiale dans l'espace image. La seconde est un *tri tardif* (ou *sort-last* en anglais) sur les fragments appartenant à un même pixel.

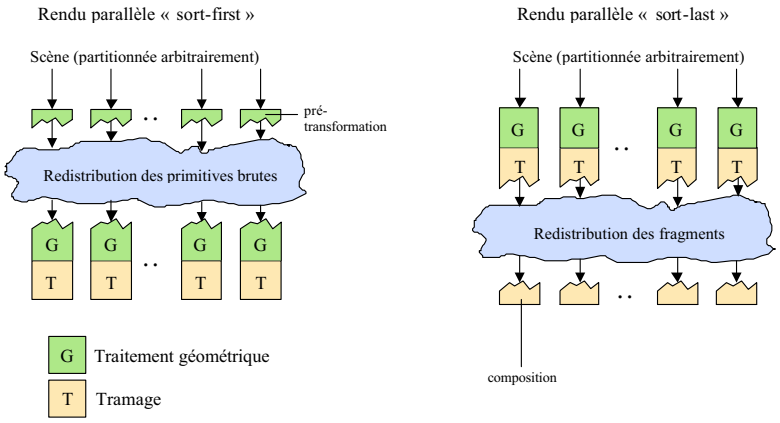


FIG. 1.17: **Stratégies de tri pour la visualisation scientifique sur grappe.** Deux types de tri existent : le *tri précoce* (ou *sort-first*) sur les primitives géométriques en fonction de leur position spatiale dans l'espace image et le *tri tardif* (ou *sort-last*) sur les fragments appartenant à un même pixel.

Ces stratégies sont ainsi utilisées pour le rendu volumique direct : lancer de rayons [Ma95, Cas06] comme projection de cellules [MC97] ; et pour l'extraction d'isosurfaces [HH92].

La mise en place de telles stratégies interactives nécessite en premier lieu un matériel de pointe constitué d'ordinateurs et de cartes graphiques performants mais aussi d'un réseau rapide diminuant les temps de transfert d'information. La disposition d'une telle infrastructure idéale ne concerne donc qu'un nombre encore limité d'industriels et de chercheurs. La gestion d'un réseau à la bande passante limitée reste un des problèmes majeurs pouvant affecter l'interactivité de telles solutions.

5.2 La localisation géographique des données

Afin de réaliser des simulations complexes, les ingénieurs et les chercheurs font appel aux supercalculateurs. Pour des raisons de coût ceux-ci se rendent rarement physiquement sur le site où se trouvent ces unités de calcul afin d'analyser les résultats. La visualisation scientifique est donc confrontée au rapatriement des résultats sur un ordinateur de bureau parfois distant de milliers de kilomètres des supercalculateurs. La garantie d'une exploration interactive des données dans ce cadre est donc fortement contrainte par la bande passante et les conditions de transmission du réseau. Le but ultime de cette problématique étant d'offrir les mêmes libertés à l'utilisateur que si les données se trouvaient entièrement sur son ordinateur ou des disques proches reliées par une connexion à haut débit de transfert.

Une première solution envisageable est de produire les images grâce au supercalculateur. Ces images sont ensuite rapatriées vers l'utilisateur en utilisant le réseau. Cette solution simple est performante. De plus, de nombreux algorithmes de *streaming* et de compression d'images comme de vidéos existent pour accélérer les taux de transfert de telles informations. Mais, il n'est pas toujours possible d'appliquer une telle méthode, les supercalculateurs pouvant n'inclure aucune carte graphique. Deux techniques sont alors principalement utilisées pour pallier à cette absence : la compression [GGS99, SR99, ILGS06] et/ou la simplification des données [THJ99, CCM+00, NE04] ; dans un but commun, celui de diminuer la quantité de données à transférer.

Ces deux solutions travaillant directement sur la géométrie ou les valeurs des résultats sont rarement adaptées aux besoins de l'utilisateur qui souhaite travailler à pleine résolution sans perte d'information[12]. Une approche alternative est l'utilisation de la multirésolution [CDFM+04]. Mais celle-ci ne semble justifier que si la machine distante est capable de générer un tel schéma et d'envoyer au fur et à mesure les données intéressant l'utilisateur.

[12]Une compression sans quantification est une solution mais les schémas proposés jusqu'à lors de compression pour les maillages irréguliers n'évoquent guère cette hypothèse sans voir leur taux de compression fortement diminué par une telle assertion.

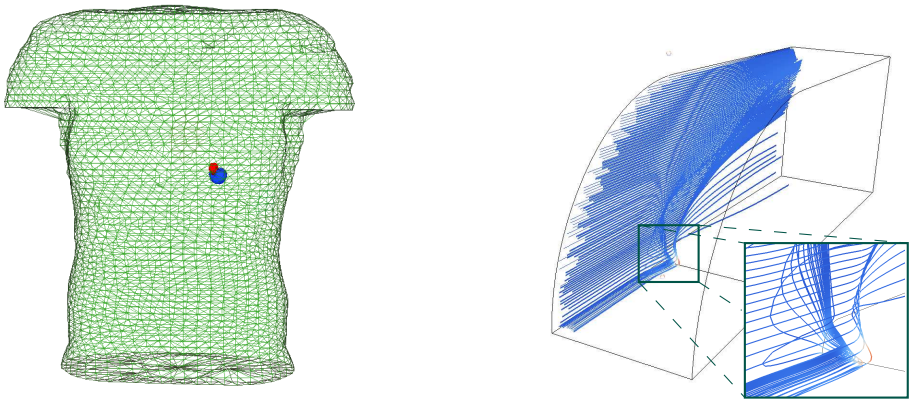


FIG. 1.18: **Exemples de la localité de zones d'intérêt compactes.** À gauche, ensemble de données *Torso*. On étudie le champ bioélectrique produit par le cœur. Le champ scalaire représente ainsi des potentiels quasi constant au sein du torse excepté autour du coeur [MJE91]. Le champ scalaire est visualisé *via* une représentation filaire de la surface et deux isosurfaces bleue et rouge représentant les bas et hauts potentiels respectivement. On remarque ainsi que l'information est localisée uniquement près du cœur. À droite, ensemble de données *Blunt Fin*. On utilise des *streamlines* et la boîte englobante des données pour visualiser le champ vectoriel. On remarque que les variations brusques en rouge (création d'un vortex) sont locales. Au sein de l'encart, un grossissement sur la région d'intérêt est fourni.

5.3 Localité de l'information

L'un des buts de la visualisation scientifique pour les grands ensembles de données de quelque nature qu'ils soient (grilles régulières, maillages irréguliers) est de mettre en évidence les régions caractéristiques des champs issus de la simulation. On nomme ces régions caractéristiques des *zones d'intérêt*. Ces zones peuvent être repérées au sein du maillage via l'utilisation d'une fonction de transfert et l'utilisation de divers rendus que nous avons détaillés en section 3.3. Une fois celles-ci identifiées, le scientifique ou l'ingénieur peut essayer de qualifier chaque région en se posant des questions comme : qu'est-ce que c'est ? D'où cela provient-il ? Comment évolue-t-elle et combien de temps persiste-t-elle (dans le cadre de données temporelles) ? La colorisation, la relation avec l'alentour spatial et temporel sont des clefs essentielles pour comprendre le phénomène puis valider la simulation ou l'expérience.

Néanmoins, la localisation de ces zones d'intérêt est un challenge au sein des grosses masses de données. En effet, celle-ci repose sur deux défis majeurs : le choix d'une "bonne" fonction de transfert et l'affichage interactif de millions de tétraèdres (évoqué précédemment). Notons que le premier défi n'entre pas dans le sujet de cette thèse mais il consiste à construire une surjection adéquate entre les espaces scalaires et l'espace des couleurs RGBA afin que les zones d'intérêt soient visibles sans que l'utilisateur ne modifie à la main lui-même cette bijection. Cette "bonne" paramétrisation de la fonction de transfert représente lorsqu'elle n'est pas (semi-) automatisée, un travail long et fastidieux afin que le rendu final soit porteur de sens pour toute une communauté. Certaines techniques ont tenté d'automatiser ce choix et le lecteur intéressé pourra se référer aux articles suivants [KD98, RSHSG00, PLB+01].

Cette difficulté de localisation est de plus renforcée par la petitesse de ces zones souvent compactes, pouvant parfois se résumer à moins de 1% de la masse de données initiale. Par exemple pour la visualisation de données médicales, Kahler *et al.* [KSH03], étudient les neurones au sein du cerveau d'une abeille – ceux-ci ne représentant que $0,8$ % des données initiales ; ou encore l'arbre vasculaire au sein du foie – celui-ci ne représentant que $3,9$ % des données initiales. On retrouve des cas similaires pour la simulation comme l'illustre la figure 1.18 à la fois lors de l'étude de champs scalaires – des potentiels dans le cas du *Torso* – ou de champs vectoriels – la vitesse dans le cas du *Blunt Fin*.

Plusieurs solutions ont déjà été proposées pour aider à la localisation de telles zones d'intérêt. Elles reposent principalement sur une assertion de base : soit l'utilisateur a une connaissance *a priori* de ce qu'il va visualiser ;

soit l'utilisateur ne projette aucune (ou peu d') information sur le maillage. Dans le premier cas, l'extraction des zones d'intérêt est automatisée *via* des précalculs ou repose sur un seuillage du champ d'attributs. Dans le second cas, une exploration de l'ensemble de données est proposée grâce à la définition d'une zone locale de raffinement représentant métaphoriquement une loupe.

Ainsi, Silver et Zabusky [SZ93] proposent d'extraire les zones d'intérêt, en choisissant une source *via* un seuil d'intérêt et une expansion par *region growing*, afin d'assurer leur suivi temporel au sein de données vectorielles temporelles. Les techniques Focus+Contexte (F+C) [VFSG06] développées pour les grilles régulières se reposent sur la segmentation des données afin d'adapter le rendu aux zones d'intérêt. Pour les plus gros ensembles de données, les techniques de *raffinement de maillage adaptatif* (AMR) développées pour les maillages uniformes [KSH03] et de multirésolution pour les maillages irréguliers [CDFM+04] permettent de réduire l'information là où celle-ci est peu pertinente tout en conservant à pleine résolution les zones d'intérêt. L'AMR est une structure calculée en prétraitement fusionnant les cellules ne contenant pas d'information essentielle. Les approches multirésolution stockent une hiérarchie d'opérations de simplification locales permettant lors de l'exploration de raffiner à la volée les zones d'intérêt.

De telles solutions restent néanmoins limitées à de petits maillages dans le cas de grilles irrégulières issues de simulation car les temps de précalculs et d'extraction restent importants.

6 Conclusion et Objectifs

La visualisation scientifique s'inscrit dans un procédé de validation de résultats issus de simulations diverses et variées concernant l'ensemble des domaines dits scientifiques comme l'astronomie, la chimie, la thermodynamique, la mécanique des fluides, la géologie, la neutronique ou la biologie. La génération interactive d'images porteuses de sens lors de la phase d'exploitation des résultats est donc essentielle afin que l'ingénieur ou le chercheur puisse réperer les zones d'intérêt et valider sa simulation avant la prise de décision finale.

Un certain nombre de ces simulations reposent sur des maillages tétraédriques générés grâce à des approches dites de *Delaunay*. Leur création, dépendant de règles mathématiques établies en géométrie algorithmique, fait de ce type de maillage de bons candidats pour l'élaboration de simulation. L'augmentation de puissance de calculs des cartes graphiques incorporant un pipeline graphique optimisé a permis au cours des années précédentes une accélération des techniques de visualisation mais non suffisant au regard de la croissance de la taille des résultats.

La visualisation scientifique reste ainsi confrontée à un problème majeur : l'exploration interactive et l'aide à la localisation rapide des zones d'intérêt face à l'augmentation sans cesse renouvelée de la taille des données issues des simulations. De précédentes approches tentent de répondre en partie à cette problématique mais ne se focalisent que sur l'interactivité ou la localisation dans des domaines précis. Ce problème reste pleinement d'actualité.

L'optique de cette thèse est de proposer, dans le cadre des grilles irrégulières tétraédriques, des solutions permettant une exploration interactive des données tout en facilitant la localisation des zones d'intérêt de manière générique, c'est-à-dire en supposant que l'utilisateur projette peu de connaissances sur les résultats – ce qui revient à ne pas spécialiser l'approche à un domaine scientifique particulier.

CHAPITRE 2

Bi-Résolution : Concepts et Précalculs

> Its specific goal is to act as a catalyst between scientific computation and scientific insight. Scientific visualization came into being to meet the ever increasing need to deal with highly active, very dense data sources.
>
> Ed FERGUSON - *Computer Graphics Career Handbook*, 1991

LA COMPRÉHENSION des résultats issus des simulations numériques est une étape cruciale dans la validation d'un modèle mathématique. La visualisation scientifique répond à cette attente en offrant des algorithmes appropriés pour afficher des images porteuses de sens pour un public, en premier lieu, averti. La taille des données est le principal facteur limitant de l'efficacité des algorithmes de visualisation. Ce domaine nécessite ainsi de continuelles améliorations en terme de performances temporelles et mémoires afin de garantir une exploration interactive pour la compréhension rapide des résultats simulés lors de leur exploitation.

Afin d'accélérer l'affichage des maillages irréguliers tétraédriques de grandes tailles, trois approches sont actuellement développées : une simplification de la géométrie du maillage en amont ; une optimisation du pipeline graphique en aval ; et une parallélisation des algorithmes sur des grappes d'ordinateurs et de cartes graphiques. Cette dernière, bien qu'incontournable au sein de la communauté de visualisation, est en dehors du cadre de cette thèse. Nous ne nous étendrons donc pas plus sur ces techniques. Néanmoins, le lecteur intéressé pourra se référer à [ABM+01, CGM+06, Cas06, VCS+07] pour plus de détails sur ce sujet. A contrario, la première approche reposant essentiellement sur des algorithmes de simplification et de multirésolution sera traitée dans ce chapitre alors que les algorithmes de rendu seront abordés au sein du chapitre 4.

En effet, pour pouvoir visualiser plusieurs dizaines de millions de tétraèdres, les progrès architecturaux des ordinateurs et des cartes graphiques, bien qu'en constant progrès, ne permettent pas de les afficher *en temps-réel* (voir chapitre 1, section 3.2). Et l'augmentation exponentielle du nombre des primitives des maillages afin de gagner en précision garantit que ce décalage entre tailles des données et puissance de calculs perdura encore

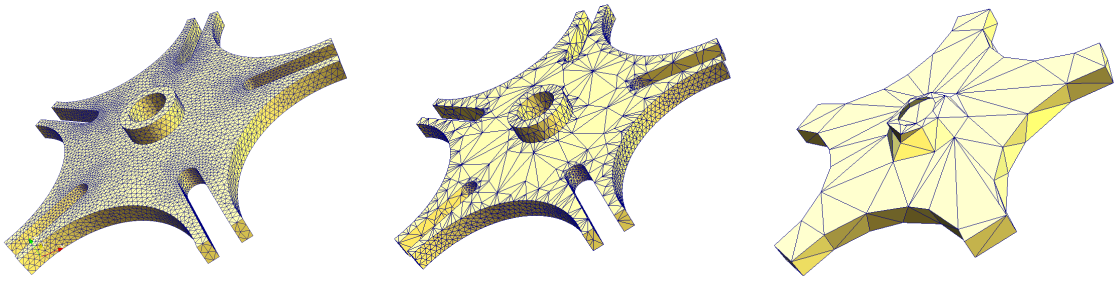


FIG. 2.1: **Simplification surfacique de la peau d'une pièce mécanique**. A gauche, la pièce est affichée à pleine résolution (18 926 faces). Au centre, la topologie et la position des sommets géométriques sont préservées lors du processus de simplification (5 420 faces). A droite, la topologie et la position des sommets ne sont pas conservées (228 faces).

pendant de nombreuses années. D'où la nécessité de traiter les maillages irréguliers afin de concilier précision nécessaire à l'exploitation et interactivité utile à l'exploration visuelle.

Une solution, inspirée des traitements réalisés sur les maillages surfaciques triangulés, est d'appliquer une simplification du maillage permettant de diminuer de manière considérable le nombre de primitives tout en conservant une erreur minimale au maillage initial. Une fois ces algorithmes développés, une approche multirésolution peut être précalculée afin d'extraire de manière dynamique des niveaux de détails (LOD pour Level of Details) qui permettront de conserver selon les besoins de l'utilisateur une précision nécessaire dans des zones spécifiées. Le nombre de primitives finales étant drastiquement diminué, ces méthodes permettent ainsi une exploration des données de manière interactive lors de la phase d'exploitation.

Dans ce chapitre, nous introduisons une nouvelle approche, constituant le socle de cette thèse, pour réaliser un schéma multirésolution qui se base uniquement sur deux niveaux de détails statiques qui sont combinés de manière dynamique en fonction d'une région d'intérêt manipulée par l'utilisateur. Cela permet ainsi de diminuer à la fois l'occupation mémoire, les temps de précalculs et d'accélérer l'extraction dynamique et le rendu. Afin de mieux cerner les tenants et les aboutissants d'une telle méthode, nous revenons dans un premier temps sur les précédents travaux réalisés en matière de simplification et de multirésolution tétraédriques dans la section 1. Nous détaillons ensuite les étapes de précalculs qui permettent lors de l'exécution une extraction dynamique de niveaux de détails au sein de la section 2, en insistant sur les objectifs que nous souhaitons atteindre (section 2.1). Ces étapes contiennent une simplification éventuelle adaptée pour les maillages tétraédriques de complexité faible en occupation mémoire (section 2.2) et le calcul obligatoire d'une partition des sommets du maillage fin (section 2.3). Enfin, nous discutons les résultats de ces étapes de précalculs en section 3. La mise en œuvre de l'extraction dynamique afin d'obtenir un maillage birésolution sera précisée dans le chapitre 3.

1 Simplification et MultiRésolution

Afin de réaliser des approches multirésolution sur des complexes simpliciaux, un ensemble de techniques de simplification hiérarchique ont été développées depuis quelques années. Ces méthodes peuvent être regroupées selon un certain nombre de critères :

- *la portée de modification* : globale (comme les approches utilisant les ondelettes) ou itérative gloutonne (essentiellement des techniques de décimation) ;
- *la préservation de la géométrie* : rééchantillonnage (avec modification des positions) ou sous-échantillonnage (conservation des positions initiales) des sommets ;
- *la préservation de la topologie* : du maillage (trous, tunnels) et/ou des données (points singuliers, vortex, sources, puits, ...) ;
- *l'évaluation de l'erreur au maillage initial* : géométrique, topologique ou basée sur les attributs ;
- *la condition d'arrêt* : basée sur l'erreur d'approximation ou sur un seuil en nombre de simplexes (habituellement sommets ou cellules).

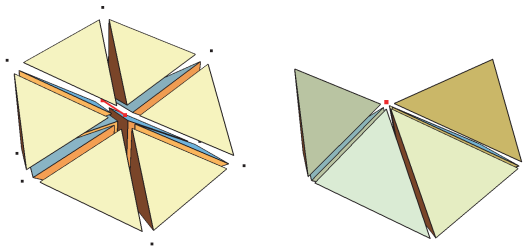


FIG. 2.2: **Décimation d'arêtes : l'opération *half-edge collapse***. À gauche, l'arête rouge va être effondrée sur le sommet rouge du fond. À droite, résultat de la décimation.

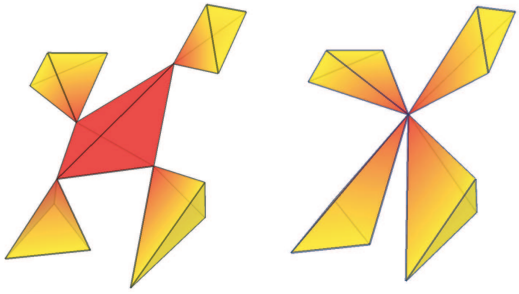


FIG. 2.3: **Décimation de tétraèdres : l'opération *tetfuse*.** À gauche, le tétraèdre rouge central va être effondré en un unique sommet : son barycentre. À droite, résultat de l'opération de décimation

Dans la suite, nous nous consacrerons uniquement aux techniques de décimation itératives et à celles qui tentent de préserver la qualité du maillage (section 1.1). Nous détaillerons ensuite les différentes techniques de multirésolution qui en sont issues (section 1.2).

1.1 Simplification

Initialement développées pour les maillages triangulaires [RB93, HDD+93, HG97] afin de répondre aux attentes des industries des jeux vidéos et de simulation de vols, les méthodes de décimation ont été rapidement généralisées aux maillages tétraédriques volumiques pour plusieurs raisons. La majorité d'entre elles n'applique pas de rééchantillonnage des sommets, *id est* que les sommets composant les maillages simplifiés correspondent à des sommets originaux ; ce qui permet ainsi de conserver les données initialement calculées aux sommets lors de la simulation. Elles préservent de plus la topologie, critère essentiel afin de préserver les formes complexes de pièces issues de simulations mécaniques comme illustré au sein la figure 2.1 où les parties échancrées ainsi que le moyeu de cette pièce ne doivent être détériorés durant le processus de simplification.

Le maillage grossier résultant pourra être utilisé dans certaines situations pour calculer une nouvelle simulation à moindre échelle en utilisant la méthode des *éléments finis*. Le processus de simplification se doit alors de garantir une certaine qualité des tétraèdres (voir chapitre 1, section 2.2) afin que les calculs d'intégration ne divergent pas. De tels tétraèdres garantissent en plus des rendus avec moins d'artefacts lors de la visualisation [UBF+05].

Nous détaillons tout d'abord les opérations locales de décimation permettant de générer un maillage tétraédrique simplifié (section 1.1.1) puis les différentes erreurs élaborées pour minimiser la différence entre le maillage initial et le maillage grossier (section 1.1.2) ainsi que les critères de qualité (section 1.1.3) avant de préciser les limitations de telles approches (section 1.1.4).

1.1.1 Décimation

La *décimation* d'un *d-simplexe* σ : sommet, arête, face ou cellule ; consiste à le remplacer par un *l*-simplexe ν de dimensionnalité inférieure ($dim\ \nu < dim\ \sigma$), habituellement un sommet. Cela revient à appliquer une surjection aux sommets v de σ afin de les transformer en sommets w de ν. Cette opération résulte non seulement à la modification de σ en ν mais aussi à celle des simplexes appartenant à l'union des étoilés des sommets v de σ (cf. définition 1.3).

Les différentes approches développées pour les maillages tétraédriques sont détaillées ci-après.

Décimation d'arêtes La décimation ou l'effondrement d'arêtes (*edge-collapse*) ont été introduits par [HDD+93]. Elle consiste à effondrer une arête $e =< u, v >$ en un sommet w défini comme une combinaison barycentrique des deux sommets d'origine. Lorsque $w \in \{u,\ v\}$, on parle d'effondrement de demi-arêtes (*half edge-collapse*). Cette dernière, illustrée en figure 2.2, est la plus utilisée. En effet, elle conserve la position des sommets originaux ce qui limite la consommation mémoire et permet de réutiliser en ces points le champ scalaire.

Elle reste la technique locale gloutonne la plus utilisée pour simplifier un maillage tétraédrique. En plus de sa simplicité, elle garantit en étudiant le *lien* (cf. Définition 1.4) de l'arête élue pour être effondrée la conservation de la topologie du maillage initial [DEGN99], voire également de sous-structures imbriquées dans le maillage initial [Viv05].

Décimation de tétraèdres Ce type de décimation a été introduit par [THJW98] en proposant une succession de contractions de trois arêtes successives au sein d'un même tétraèdre. Néanmoins, l'utilisation des arêtes impose un surcoût mémoire (de l'ordre du nombre de tétraèdres dans la majorité des cas) afin de réaliser leurs effondrements. Pour pallier à cet inconvénient, [CM02] propose une nouvelle opération appelée *tetfuse* qui agit directement sur les tétraèdres. Elle effondre les cellules en un sommet par exemple leur barycentre comme illustré dans la figure 2.3 où le tétraèdre central est effondré sur lui-même. Notons que, contrairement à l'effondrement de demi-arêtes, cette décimation impose la création de nouveaux sommets dont les valeurs scalaires associées doivent être calculées par interpolation.

Cette décimation provoque au minimum la suppression de onze tétraèdres en une seule étape. Son principal intérêt est donc d'accélérer grandement le processus mais de permettre aussi un meilleur ratio de simplification par rapport au maillage initial. Néanmoins, afin de conserver la surface de bord du maillage, les tétraèdres qui pourraient la modifier, *id est* ayant au moins un sous-simplexe appartenant au bord, ne peuvent être effondrés. Une amélioration est proposée dans [CM03] en utilisant une erreur métrique quadrique (QEM) pour améliorer la position géométrique du sommet d'effondrement. De plus, la topologie du maillage est conservée en prenant garde aux inversions de sommets (*flipping*), notamment au bord. Les tétraèdres formant le bord du maillage peuvent ainsi être inclus dans le processus.

1.1.2 Erreurs d'approximation

La simplification d'un maillage par décimation ne peut être réalisée sans être guidée. Afin de conserver les propriétés du maillage initial, trois principales types d'erreurs ont été introduites : erreur géométrique, erreur topologique et erreur sur les champs associés issus de la simulation. On cherche lors du processus de simplification à minimiser ces erreurs afin de conserver un maillage grossier proche – au sens de la définition des erreurs – de l'original. De plus, une pondération peut être réalisée afin de favoriser un type d'erreur plus qu'un autre en fonction des besoins de l'utilisateur.

Concrètement, ces trois erreurs permettent d'associer une erreur à chaque simplexe. À chaque étape, le simplexe avec la plus faible erreur est décimé et son effondrement entraîne la mise à jour des simplexes voisins modifiés par cette opération ainsi que leurs erreurs associées. Le processus de simplification continue jusqu'à ce que la condition d'arrêt soit vérifiée. Cette condition peut être un nombre fixe de cellules ou une valeur maximale d'erreur d'approximation entre les deux maillages.

Erreur Géométrique Cette erreur introduit plusieurs notions. Tout d'abord, elle permet de vérifier que la décimation d'un simplexe n'introduit pas d'inversion de tétraèdres, de *slivers*, voire d'autointersection [CCM+00]. Elle permet aussi de calculer la différence spatiale par rapport au maillage initial. Une distance de *Haussdorf* peut ainsi être utilisée pour calculer l'erreur liée aux modifications des surfaces de bord.

Une autre erreur possible utilisée initialement pour les maillages surfaciques [GH97], est l'erreur métrique quadrique. Elle extrait une mesure géométrique pour chaque sommet en calculant la somme des distances de ce point vers un ensemble de plans dépendant du maillage initial. La différence entre la mesure initiale et celle après transformation permet de définir une erreur. Cette erreur est efficace et simple à stocker car elle peut s'exprimer comme une matrice carrée 4×4.

De plus, les simplexes appartenant à la surface subissent un traitement particulier afin de conserver les propriétés topologiques et géométriques de celle-ci.

Erreur Topologique Cette erreur permet de conserver la topologie d'un maillage volumique. Elle se base sur des calculs d'intersections de *liens* associés aux simplexes (ici des arêtes) effondrés. Pour plus de précision, le lecteur peut se référer à [NE04, Viv05, VBHHar].

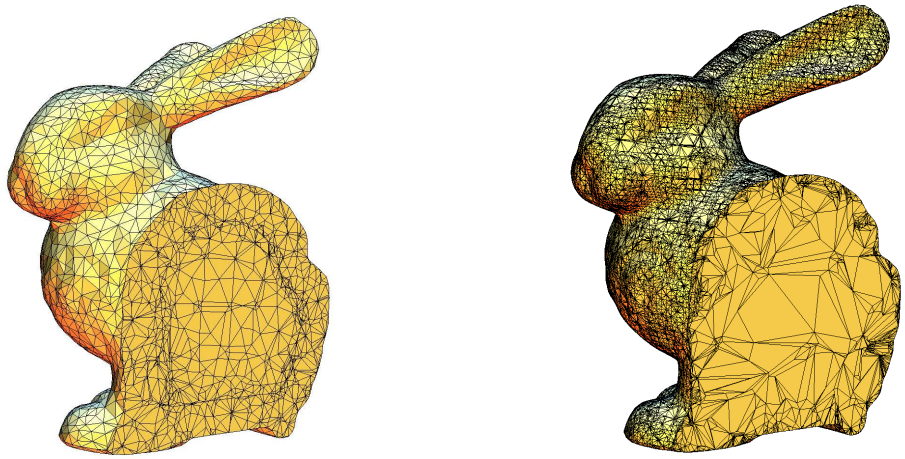


FIG. 2.4: **Simplification volumique d'un maillage tétraédrique**. Le maillage initial est composé de 115 378 sommets. À gauche, le maillage est simplifié en utilisant l'algorithme de [CDM04] conservant la qualité des tétraèdres (10 701 sommets). À droite, le maillage est simplifié au maximum en utilisant l'algorithme de [CCM+00] (56 355 sommets). La qualité des tétraèdres diffère fortement entre ceux se trouvant sur la surface de bord et ceux à l'intérieur.

Erreur sur les Attributs Les attributs sont les champs scalaires, vectoriels ou tensoriels issus de la simulation. Afin de ne pas détériorer les résultats, ceux-ci doivent recevoir une attention toute particulière lors du processus de simplification. Une simple métrique d'erreur basée sur une distance euclidienne peut être utilisée [CCM+00]. Des erreurs plus élaborées ont aussi été proposées comme la définition d'une spline sur les tétraèdres du maillage [THJ99] ou une *erreur métrique de densité* [VGVW99] pondérant la variation du champ scalaire par la variation du volume des tétraèdres modifiés.

1.1.3 Conserver la qualité du maillage

Les erreurs précédentes ne prennent pas en compte la qualité des tétraèdres même si l'erreur géométrique tend à limiter les *slivers*. Par exemple, dans [NE04], un coefficient permet de pondérer l'importance de l'erreur métrique quadrique afin d'influencer la qualité des tétraèdres grossiers. Or l'aspect des tétraèdres est primordial pour à la fois permettre un nouveau calcul de simulation sur le maillage simplifié et garantir un rendu limitant les artefacts.

Une approche pour répondre à cette problématique a été proposée par [CDM04]. Elle utilise des opérations locales sur les tétraèdres, principalement des contractions d'arêtes mais aussi des échanges d'arêtes, des déplacements ou insertions de sommets. Ces opérations permettent de garantir que les tétraèdres issus du processus de simplification vérifient une mesure de qualité. Elle est définie sur trois paramètres géométriques : l'angle solide (cf. définition 1.8), le volume et les arêtes.

Définition 2.1 Erreur de Qualité : Soit Σ le maillage tétraédrique, τ le nombre de tétraèdres que l'on souhaite atteindre en fin de simplification. On définit tout d'abord le volume idéal $V_i = \frac{V(\Sigma)}{\tau}$ où $V(\Sigma)$ est le volume du maillage, et la taille de l'arête idéale $l_i = \sqrt[3]{V_i}$. L'erreur de qualité Q est définie ainsi :

$$Q(t) = \sqrt[3]{Q_{angle}(t) + Q_{volume}(t) + Q_{arete}(t)}$$

$$Q_{angle}(t) = \frac{angle_solide_minimum(t)}{0,55}$$

$$Q_{volume}(t) = clamp(\frac{V(t)}{V_i})$$

$$Q_{arete}(t) = clamp(\frac{5\ l_i}{plus_longue_arete(t)})$$

	tétraèdres initiaux	ratio	temps
[CM02]	1 499 160	36,21 %	187,23 *s*
[CDM04]	1 596 000	0,06 %	29 *min* 01 *s*
[UBF+05]	1 499 160	12,0 %	36,03 *s*
[VCL+07]	13 980 162	10,0 %	40 *min* + 244,42 *s*

TAB. 2.1: **Temps de simplification de maillages tétraédriques des précédentes approches**. Les données sont extraites des articles référencés. Le pourcentage du maillage restant par rapport au nombre de tétraèdres originaux ainsi que les temps de simplification sont renseignés pour les plus récentes méthodes de simplification. Pour [VCL+07], le temps de simplification est composé de deux étapes : la transformation du fichier initial, puis le processus de simplification en *streaming* basé sur le nouveau fichier.

La figure 2.4 illustre la différence obtenue en utilisant cet algorithme [CDM04] et celui de [CCM+00]. La différence sur la forme des tétraèdres est assez disparate entre l'intérieur et la surface de bord pour ce dernier alors que le premier conserve une certaine régularité à la fois dans l'échantillonage et dans la forme. On note de plus que le premier algorithme permet de simplifier plus fortement le maillage initial.

1.1.4 Limitations

Bien que les algorithmes de simplification de maillages tétraédriques actuels soient performants, la majorité d'entre eux repose sur des opérations de décimation locales qui compliquent les généralisations en mémoire externe. En effet, la nécessité de la connaissance du *lien* des éléments effondrés impose des restrictions lors du prédécoupage du maillage à cause des relations de connectivité nécessaires à leur bon déroulement. Ainsi, les derniers algorithmes proposés reposant sur ces approches locales ne proposent pas pour la plupart de simplification pour des maillages ayant plus d'un million et demi de tétraèdres – cf. tableau 2.1.

Néanmoins, afin de traiter les maillages issus des simulations numériques de taille toujours croissante, des simplifications en mémoire externe deviennent plus que nécessaires, prenant de surcroît en compte la qualité des tétraèdres. Une première étape vers le traitement de telles masses de données a été récemment proposée en utilisant une simplification par *streaming* [VCL+07]. Cette méthode n'impose aucune limite de taille sur les maillages tétraédriques. Par exemple, la simplification d'un maillage composé d'un milliard de cellules en un maillage de douze millions de tétraèdres est réalisée en dix heures. Pourtant, cette approche a deux limitations majeures. Premièrement, la qualité des tétraèdres n'est pas pris en compte. Deuxièmement, l'algorithme nécessite une réorganisation du fichier utilisée en lecture. Celle-ci réalise en quarante minutes pour un maillage composé de quatorze millions de tétraèdres ce qui est un facteur limitant – cf. dernière ligne du tableau 2.1.

Afin de répondre à ces problématiques, nous proposons une amélioration de la méthode développée par [UBF+05] détaillée en section 2.2 afin de l'étendre à la simplification de plus gros ensembles de données tout en conservant la qualité des maillages grossiers résultants, sans aucun précalcul nécessaire sur le fichier de données.

1.2 MultiRésolution

Les maillages issus de la simplification ne sont pas utilisés directement lors de la phase de visualisation. Bien que conservant, grâce aux erreurs d'approximation, les caractéristiques du maillage et de ses champs associés, une partie de l'information peut être manquante dans des zones d'un intérêt particulier. En effet, lors de l'exploitation visuelle des résultats, le nombre de cellules affichées à l'écran peut être réduit à quelques milliers de tétraèdres afin que celle-ci soit temps-réel. Cela peut impliquer un relâchement d'une ou plusieurs erreurs d'approximation au cours de la simplification et donc provoquer une dégradation des données. Ce qui aura pour conséquence de nuire à l'analyse des données et à leur compréhension.

La granularité au sein des zones critiques (extrema de potentiels, vortex de champs vectoriels, ...) est capitale à la fois pour les identifier au cours de l'exploration puis pour les analyser. Comme le niveau de détails initial ne peut être affiché interactivement et que le niveau de détails final peut être incorrect à cause de détériorations géométriques, topologiques ou sur les variations des champs d'attributs, des algorithmes dit *multirésolution* ont été développés afin de répondre aux deux problématiques antonymes des ingénieurs réalisant la visualisation des simulations : l'exploration interactive et la conservation de la granularité initiale au sein des zones d'intérêt.

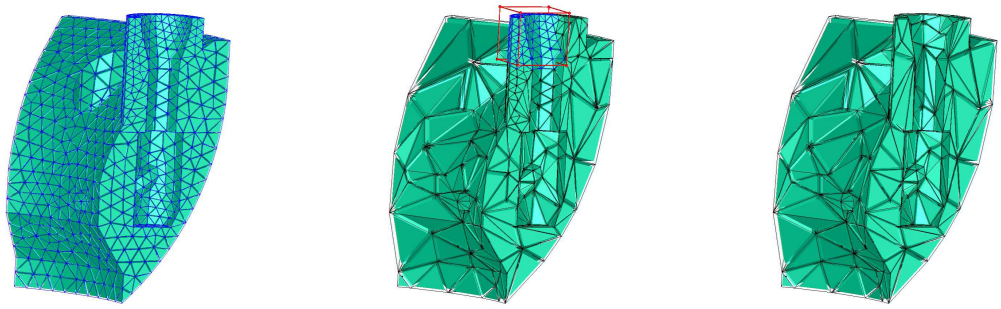


FIG. 2.5: **Multirésolution volumique d'un maillage tétraédrique**. À gauche, le maillage original. À droite, le maillage simplifié. Au milieu, une fine région d'intérêt au sein de la boîte englobante est extraite en utilisant MT [DFMP00]. A l'extérieur, les tétraèdres restent au niveau le plus grossier. La transition entre les deux niveaux de détails est extraite à la volée à partir des opérations locales en parcourant la structure de données.

La figure 2.5 illustre une extraction dynamique d'un niveau de détails au sein d'une zone géométrique précise définie par les contours d'une boîte englobante. Le maillage initial à haute résolution ainsi que le maillage final à basse résolution sont aussi affichés afin de fournir un comparatif. La précision la plus haute est extraite au sein de la zone et une continuité est assurée avec le maillage grossier en dehors de cette zone.

Nous détaillons dans un premier temps le principe des approches multirésolution (section 1.2.1) avant de décrire les structures de données utilisées (section 1.2.2) et d'analyser les limitations des solutions actuelles (section 1.2.3)

1.2.1 Principe

Les opérations d'effondrement résultent en une séquence linéaire de niveaux de détails. Grâce à un codage des dépendances entre ces opérations, les algorithmes de multirésolution permettent de généraliser les niveaux de détails dynamiquement, en simulant un changement dans l'ordre d'effondrement des simplexes. Cela permet notamment de conserver les détails dans une zone arbitrairement choisie par l'utilisateur.

Afin de mettre en place un système de visualisation efficace, plusieurs challenges doivent être relevés. Premièrement, la construction d'une structure de données efficace en temps d'accès et en mémoire est capitale afin de garantir l'extraction interactive d'un niveau de détails. Nous revenons plus précisément sur ce point dans la sous-section suivante. Deuxièmement, des erreurs dynamiques doivent être définies afin de répondre aux besoins précis de l'utilisateur qui explore les résultats de la simulation. De nombreux degrés de liberté doivent être pris en compte afin d'offrir le meilleur outil de visualisation dans tous les cas possibles d'utilisation.

Ainsi, les erreurs d'approximation dynamiques peuvent être de plusieurs types [CDFM+04] :

- *basée sur le point de vue de l'utilisateur* : les tétraèdres visibles seront raffinés alors que ceux invisibles seront simplifiés.
- *basée sur une localisation spatiale* : Cette erreur spatiale est utilisée pour l'exploration interactive de gros maillages de données. En effet, elle permet de restreindre l'erreur précédente à une zone d'intérêt déplacée par l'utilisateur au sein de la simulation, ce qui réduit de manière drastique le nombre de tétraèdres affichés tout en gardant une résolution élevée dans le focus restreint.
- *basée sur une valeur ou un intervalle du champ d'attributs* : les tétraèdres comprenant ces valeurs seront raffinés. Cela permet par exemple d'obtenir une surface isovaleur à une haute résolution tout en conservant des tétraèdres grossiers hors de sa zone de définition.
- *basée sur la fonction de transfert* : les tétraèdres les plus transparents sont simplifiés.

La figure 2.6 illustre deux de ces erreurs d'approximation dynamique : basée sur la localisation spatiale et basée sur une isovaleur.

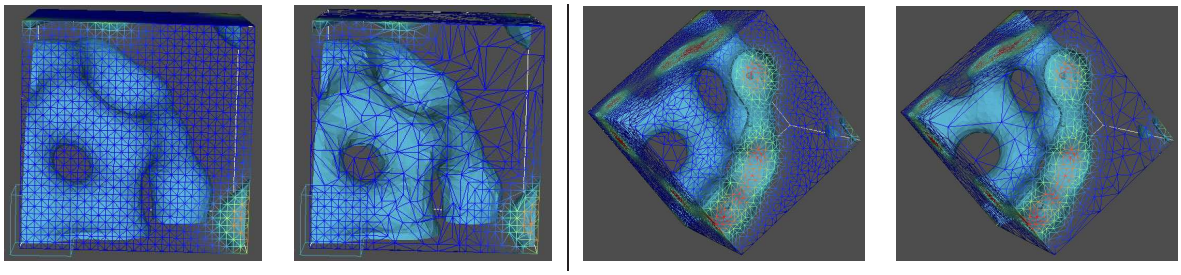


FIG. 2.6: **Erreurs d'approximation dynamique pour l'extraction de niveau de détails** provenant de [DDFMP02]. À gauche, une extraction d'une zone géométrique au sein de la boîte englobante. À droite, une extraction basée sur une isovaleur. Dans les deux cas, le maillage initial est fourni afin de mettre en évidence les zones de raffinement et de simplification.

1.2.2 Structures de données

Afin de réaliser l'extraction dynamique d'un niveau de détails, il est nécessaire de stocker en mémoire l'ensemble des opérations de décimation qui ont permis de simplifier le maillage. Concrètement, une mise-à-jour d'un maillage tétraédrique Σ consiste à remplacer un ensemble de cellules Σ_1 de Σ par un autre ensemble Σ_2 de telle sorte que le résultat soit toujours un maillage. Ce besoin implique que les dépendances entre les opérations de décimations doivent aussi être sauvegardées. En effet, les décimations ne peuvent être faites de manière indépendante car cela pourrait introduire des violations géométriques qui résulteraient en un niveau de détails qui ne serait plus un complexe simplicial sans intersection géométrique : on parle alors de maillage non *conforme*.

Les structures de données multirésolution peuvent se regrouper en des catégories distinctes :
- les structures de données explicites qui encodent les tétraèdres au sein du niveau de détails ;
- les structures de données implicites qui encodent les opérations locales qui permettent d'extraire le niveau de détails ;
- les structures de données reposant sur une hiérarchie de segments.

L'ensemble de ces méthodes encode de plus les dépendances entre les opérations de décimation afin de garantir pour toute extraction un maillage dit *conforme*. Ces dépendances sont stockées soit dans un graphe sans cycle direct [DFMP00] soit dans une *forêt d'arbres binaires* [CDFMP00].

Graphe Direct Acyclique et Forêt d'Arbres Binaires Un *DAG (Directed Acyclic Graph)* est un graphe direct sans cycle, c'est-à-dire que pour tout sommet v du graphe, il n'existe aucun chemin partant de v qui revient en v. Ce DAG permet de stocker les relations de dépendances entre les différents tétraèdres. Chaque nœud du graphe correspond à une opération locale de décimation. Les arêtes indiquent les dépendances – cf figure 2.7. Néanmoins cette structure peut se révéler coûteuse lorsque le nombre d'opérations devient conséquent.

Une structure sous forme de forêts d'arbres binaires permet de limiter l'espace de stockage. Un nœud fils est alors le sommet issu de l'effondrement de ses deux nœuds pères.

Dans les deux cas, les arêtes représentant des dépendances seront appelées arcs de dépendances et la structure de données en elle-même un graphe de dépendances.

Lors de l'extraction dynamique, la structure de données choisie est parcourue le long d'un front dit *actif*. Ce front regroupe l'ensemble des nœuds permettant l'extraction des tétraèdres qui composent le maillage courant. En partant de ces nœuds, et en se basant sur les erreurs d'approximation dynamiques, le parcours de la structure de données selon les arcs de dépendances va permettre la construction de maillages *conformes*. En effet, un nœud ne pourra être inclus dans le front actif que si ses nœuds pères appartiennent au front. Si ce n'est pas le cas, cela veut dire que des opérations nécessaires au bon déroulement de l'effondrement n'ont pas été réalisées impliquant l'absence de tétraèdres nécessaires au bon déroulement de l'opération. Il faudra donc forcer les opérations liées aux nœuds pères afin de rendre cette opération valide. A l'inverse, un nœud du front actif est enlevé si et seulement si tous ses nœuds fils n'appartiennent pas au front actif.

Afin d'obtenir les meilleurs taux d'extraction, le parcours du graphe de dépendances est réalisé en utilisant deux piles à priorité sur les nœuds, l'une permettant de stocker les candidats pouvant être ajoutés, l'autre ceux

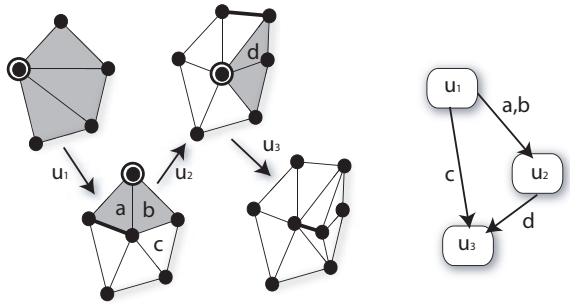


FIG. 2.7: **Graphe Direct Acyclique pour la multirésolution** issu de [DDFMP02]. Séquence de mises-à-jour d'un maillage *via* des opérations locales de dédoublement de sommets $s = (u_1, u_2, u_3)$. La partie Σ_1 modifiée par chaque opération u_i est grisée et est transformée en un autre sous-complexe simplicial Σ_2. Le sommet dédoublé est mis en évidence par un double anneau noir et l'arête issue de ce dédoublement est plus épaisse. A droite, le *DAG* issu de ces mises-à-jour. Chaque nœud représente une mise-à-jour. Les arcs représentent les dépendances entre elles. Chaque arc est de plus labellé par les triangles qui causent les dépendances.

pouvant être enlevés [CDFM+04].

Structures Explicites Tous les tétraèdres [DDFMP02] pouvant être créés ou supprimés lors de ces extractions sont stockés explicitement au sein de tableaux regroupant les positions géométriques des sommets et la connectivité comme au sein d'une structure de données indexée (voir chapitre 1, section 2.3).

Cette structure coûte $3,5$ fois plus de mémoire qu'une structure de données indexée représentant le maillage le plus fin. La taille mémoire de cette structure de données est donc sa limitation principale, ce qui a favorisé la création de structures de données implicites.

Structures Implicites Les structures de données implicites représentent les opérations de décimation au lieu des tétraèdres. Elles reposent soit sur les contraction d'arêtes [CDFM+04], soit sur les contraction de demi-arêtes [DDF02], soit sur l'opération inverse : le dédoublement de sommets [DDFMP01, SS05]. En fonction du type de l'opération, on encode au sein des nœuds du graphe de dépendances soit des arêtes, soit des sommets.

Ces structures de données essaient de minimiser la taille mémoire utilisée par une structure multirésolution afin de permettre la visualisation de maillages qui n'aurait pas pu l'être à leur résolution initiale. Ainsi, contrairement aux structures de données implicites, seules les informations nécessaires à la réalisation des opérations sont stockées, de manière à retrouver facilement la géométrie et la connectivité des nouveaux éléments. De nombreuses optimisations ont été réalisées afin de connaître l'ensemble de ces informations de manière explicite au moment opportun tout en minimisant les informations nécessaires. Ces optimisations néanmoins ont un coût et augmentent drastiquement les temps de précalculs.

Le tableau 2.2 résume les différents gains mémoire obtenus ainsi que le temps de précalculs pour l'ensemble de ces méthodes. Malheureusement la majorité de ces approches n'indiquent pas le temps de précalculs nécessaire pour mettre en place ces hiérarchies. On peut seulement dire qu'il est proportionnel au coût de simplification.

Hiérarchie de Segments Afin de s'abstraire de la suite ordonnée des opérations locales de décimation, [SS06] propose d'appliquer aux maillages tétraédriques le principe des *multi-triangulations* [Pup96].

L'idée est de réaliser une succession de partitions d'un même maillage. Chaque partition est composée de *segments*. Le nombre de segments diminue entre chaque partition successive de façon à n'obtenir à la fin qu'un unique segment. La figure 2.8 illustrent une succession de neuf partitions, chaque zone d'une couleur représente un segment de la partition.

Le respect des interdépendances entre segments de deux partitions consécutives garantit l'extraction d'un maillage *conforme*. Ces dépendances sont encodées dans un *DAG*.

La hiérarchie finale utilise en moyenne $455n$ octets où n est le nombre de sommets dans le maillage initial à haute résolution. Ce qui représente $3,9$ fois la taille mémoire utilisée pour représenter le maillage fin avec une structure indexée. C'est donc plus qu'une structure explicite. Néanmoins, une phase de compression utilisant

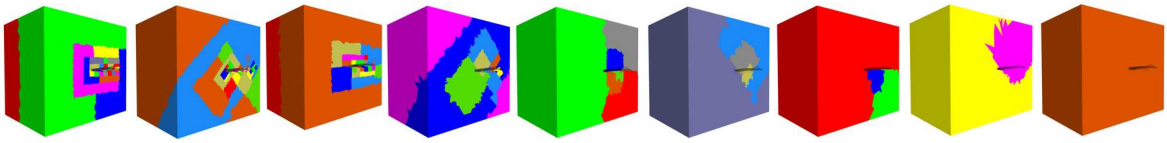


FIG. 2.8: **Hiérarchie de segments sur un maillage tétraédrique** provenant de [SS06]. Séquence des différents segments extraits à chaque niveau de la résolution la plus fine à gauche à la résolution la plus grossière (= 1 segment) à droite.

	taille mémoire	temps précalcul	
	n = # sommets	500 K tets	1.500 K tets
structure indexée	116 n	-	-
structure indexée avec adjacences	222 n	-	-
[DDFMP01]	56 n	*non communiqué*	
[DDF02]	33 n	*non communiqué*	
[CDFM+04]	40 n	13 min	35 min
[SS05]	$(37-41)\,n$	14,3 min	23,5 min

TAB. 2.2: **Occupation mémoire et temps de précalculs pour les structures implicites**. Soit n le nombre de sommets du maillage Σ. La taille mémoire des différentes structures de données est indiquée en fonction de n en octets. Les structures initiales permettant de stocker un maillage monorésolution sont aussi indiquées comme référence. Les temps de précalculs pour extraire ces structures de données sont exprimées en minutes. Deux tailles sont indiquées : 500 mille et $1,5$ millions tétraèdres.

la *Cut-Border Machine* [GGS99] permet de réduire ce coût à $103n$ ce qui est un peu moins qu'une structure indexée pour le maillage de plus haute résolution.

Les temps de précalculs, quant à eux, s'échelonnent entre 10 min pour 500 mille tétraèdres, 20 min pour 1,5 millions tétraèdres et jusqu'à 1h12min pour près de 14 millions de tétraèdres.

1.2.3 Limitations

Les approches multirésolution actuelles recherchent à répondre à deux problématiques antonymes : la minimisation de l'occupation mémoire pour stocker la hiérarchie des niveaux de détails et l'extraction dynamique rapide de ces niveaux de détails.

Concernant l'occupation mémoire, les structures implicites sont les meilleures candidates car elles permettent de réduire considérablement l'espace utilisé pour le stockage : jusqu'à 3 fois moins que la place mémoire occupée par une structure indexée sur le maillage le plus fin. Néanmoins, la construction de telles hiérarchies *via* ces structures optimisées demande un temps non négligeable de précalculs et peu de gros ensembles de données ont été testés. En effet, les processus de simplification utilisés reposant sur des opérations de décimation locales sont gourmands en place mémoire. De nombreuses relations d'adjacences sont nécessaires afin de calculer les erreurs topologiques. Un compromis mémoire/temps peut être trouvé mais pour les grands maillages, ces calculs restent chronophages.

L'approche se basant sur la hiérarchie de segments, et ainsi passant outre les opérations locales, est un premier pas vers une amélioration des temps de précalculs et aussi de la place mémoire nécessaire lors du processus de création. Néanmoins, la place occupée par la structure de données résultante est énorme. D'où la nécessité d'une compression qui ajoute là encore un temps non négligeable à l'étape de précalculs. On notera tout de même que pour un maillage composé de moins de 14 millions de tétraèdres – ce qui est un maillage de taille courante de nos jours ; le temps de précalculs est supérieur à une heure.

Le temps de précalculs est donc un premier facteur limitant pour ces approches multirésolutions.

De plus, toutes ces approches ont un autre inconvénient majeur illustré au sein de la figure 2.9. Ces extractions ont été obtenues en utilisant *MT* [CDFM+04]. On voit clairement qu'une zone assez conséquente est raffinée en dehors de la zone locale d'intérêt définie par les utilisateurs (plus de la moitié dans les deux cas). Cela est dû aux dépendances entre les différentes opérations de décimation. En effet, la création d'un niveau

FIG. 2.9: **Limitations de la multirésolution basées sur les opérations locales**. Ces deux extractions illustrent le rôle des dépendances entre les opérations de décimation. Malgré une zone locale d'intérêt réduite, l'extraction à un haut degré de granularité est faite sur plus de la moitié du maillage afin de garantir la conformité du maillage.

Méthodes	Extraction (Milliers à la seconde)
[CDFM+04]	330
[SS05]	150
[SS06]	500 *(hiérarchie brute)* 300 *(avec décompression)*

TAB. 2.3: **Flux d'extraction des primitives pour la mise à jour des niveaux de détails**. Le nombre de tétraèdres pouvant être extraits par seconde est indiqué pour les trois dernières approches multirésolution. Il varie entre 150 à 500 milliers de tétraèdres à la seconde.

de détails est fortement lié à l'algorithme de simplification. Dans certains cas, le raffinement d'une zone locale entraînera, *via* les relations de dépendances, le raffinement d'une région bien plus grande. Ce qui n'est pas acceptable si on veut garantir la localité de l'approche.

L'approche utilisant des segments restreint la zone la plus fine d'extraction aux segments les plus fins. Une zone locale telle que définie par les approches précédentes entraînera un raffinement plus local *id est* ne provoquant l'affichage que des segments les plus fins intersectant la zone locale. Ce qui rend cette approche plus intéressante dans le cadre d'une définition d'une zone locale d'intérêt. Néanmoins, les dépendances entre les segments peuvent là encore introduire des raffinements superflus afin de garantir la *conformité* du maillage dynamique au niveau des jonctions entre les segments fins au sein de la région et ceux se trouvant à l'extérieur. Ces contraintes peuvent donc là encore extraire plus de tétraèdres que nécessaire ce qui ralentira fortement l'extraction des niveaux de détails.

Concernant l'extraction dynamique des niveaux de détails, les quelques temps de mise-à-jour disponibles dans la littérature sont indiqués dans la table 2.3. Les extractions varient entre 150 mille à 500 mille tétraèdres à la seconde. Ces temps garantissent une extraction interactive de petites zones locales dans des maillages de petites tailles (quelques millions de tétraèdres). Néanmoins, si les extractions varient drastiquement entre deux images c'est-à-dire que la zone locale d'intérêt effectue un grand déplacement ou que le point de vue est inversé ; ou si les maillages deviennent plus conséquents, ces temps d'extraction ne pourront plus garantir une exploration des grands maillages de données de manière interactive.

Dans la suite de ce chapitre, nous proposons une nouvelle technique d'extraction de niveaux de détails basée uniquement sur deux résolutions. Cette technique ne reposant pas sur les opérations de décimation permet de diminuer fortement les temps de précalculs nécessaires afin de construire la hiérarchie. De plus, elle garantit une extraction locale de la zone précise comme le permet les hiérarchies de segments. Enfin, comme nous le montrons au sein du chapitre 3, elle permet des temps d'extraction des niveaux de détails plus rapides.

2 Bi-Résolution : les étapes de précalculs

2.1 Objectifs

Afin de répondre aux limitations des algorithmes de simplification et des schémas de multirésolution tout en conservant leurs qualités essentielles, nous proposons une nouvelle approche, nommée *BiRes*, qui tend à diminuer les temps de précalculs ainsi qu'à simplifier les structures de données nécessaires.

Pour cela, on cherche dans un premier temps à établir une surjection entre deux maillages à deux résolutions différentes : un maillage à haute résolution dit *fin* et un maillage à basse résolution dit *grossier* ; représentant la même simulation. Ces deux résolutions correspondent aux deux niveaux de détails extrêmes générés par les approches multirésolution. La surjection établit un lien entre les sommets fins et les sommets grossiers, résultant au calcul d'un partitionnement du maillage fin sous la contrainte du maillage grossier.

Nous détaillons ainsi au sein de ce chapitre uniquement les étapes nécessaires à l'élaboration d'une telle surjection. Son utilisation dans le cadre de l'extraction d'un maillage birésolution sera expliquée au sein du chapitre 3 suivant.

Le calcul de cette surjection nécessite en premier lieu un maillage simplifié de la simulation. Il est usuel qu'un tel maillage soit produit lors de l'étape de discrétisation de l'espace (voir chapitre 1) en parallèle du maillage haute résolution. Néanmoins, dans l'hypothèse où un maillage grossier n'aurait pas été produit, nous proposons dans un premier temps au sein de la section 2.2 un algorithme de simplification. Cet algorithme garantit une qualité du maillage généré et a été élaboré de manière à traiter des volumes de données plus grands que la majorité des précédents algorithmes.

Nous détaillons ensuite le calcul de la surjection. Le problème dit de partitionnement et les objectifs de la partition dans le cadre de notre approche sont introduits en section 2.3. Nous énonçons ensuite un critère de validité topologique de la partition en section 2.4. Enfin nous proposons et discutons un ensemble de solutions pour la définition de la surjection : l'une reposant sur la contraction d'arêtes (section 2.5) et l'autre reposant sur des critères géométriques (section 2.6).

2.2 Création d'un maillage grossier

Dans l'hypothèse où un maillage à basse résolution n'ait pas été créé, nous proposons au sein de cette section une extension d'une précédente méthode de simplification pour les maillages tétraédriques garantissant certains critères de qualité. Dans un premier temps, nous décrivons la précédente approche (section 2.2.1) puis détaillons les modifications apportées (section 2.2.2) avant de discuter et présenter les résultats (section 2.2.3).

2.2.1 Précédente approche

Afin de diminuer drastiquement les cellules d'un maillage tout en conservant les points critiques du champ initial généré par la simulation, [UBF+05] propose la méthode de simplification suivante. Il découpe le processus de simplification en trois parties indépendantes :

- *Extraction de la surface de bord et sa simplification.* En premier lieu, une distinction est faite entre les sommets appartenant à la surface de bord et ceux appartenant à l'intérieur. La surface extraite lors de ce processus est ensuite simplifiée en utilisant un algorithme de simplification spécialisé pour les surfaces triangulées.
- *Extraction des points critiques.* Les sommets intérieurs sont plongés dans un *Kd-tree* afin de les échantillonner par intervalle scalaire et de regrouper les sommets proches dans l'espace des attributs. Le niveau de granularité du *Kd-tree* définit le nombre de sommets intérieurs conservés. Plus précisément, dans chaque feuille de la structure de données, le sommet le plus proche de la moyenne scalaire sera conservé.
- *Reconstruction d'un maillage de Delaunay contraint.* La surface simplifiée de bord ainsi que les points critiques sont utilisés afin de construire un maillage de Delaunay contraint qui assure une certaine qualité des tétraèdres (cf. chapitre 1, section 2.2). Ce processus ajoute des points dits *de Steiner* qui n'auront pas de valeurs scalaires associées. Celles-ci seront extraites par interpolation linéaire en utilisant le *Kd-tree*.

L'algorithme, que nous proposons dans cette section, reprend cette méthodologie. Nous le détaillons plus précisément ci-après.

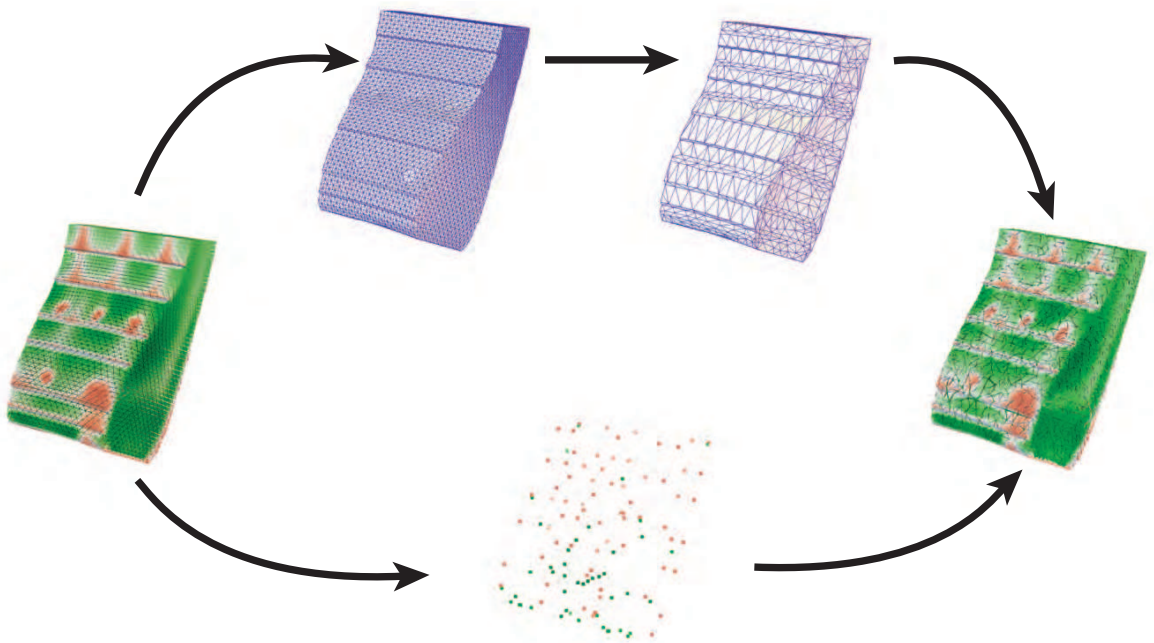


FIG. 2.10: **Étapes de la simplification des maillages tétraédriques :** Maillage fin à gauche (*Comb* : 215 040 tétraèdres) représenté en tétraèdres écorchés avec une colorisation en fonction du champ scalaire. Au centre, décomposition du maillage fin en deux sous-ensembles distincts. En haut, extraction de la surface de bord et sa simplification ; en bas, extraction des points critiques. Les points sont coloriés en fonction de leur valeur scalaire. Ces deux ensembles sont utilisés pour générer le maillage grossier de droite via une CDT (38 115 tétraèdres).

2.2.2 Algorithme de simplification

Les goulots d'étranglements mémoire de la méthode proposée par [UBF+05] ont été identifiés puis remplacés par des traitements limitant l'usage de la mémoire vive afin de permettre le traitement de plus grands ensembles de données. Pour plus de clarté, nous précisons les modifications que nous avons apportées à l'algorithme original.

Le processus de simplification est découpé par les trois mêmes étapes indépendantes, illustrées par la figure 2.10. La surface de bord est tout d'abord déterminée et simplifiée. Les points critiques du champ des attributs (ici scalaire) sont ensuite extraits afin de préserver les variations au sein du maillage grossier. Enfin, un maillage tétraédrique grossier est généré via une tétraédralisation contrainte de Delaunay (CDT) (cf. définition 1.17) ayant pour contraintes la surface simplifiée et les points critiques.

Extraction et Simplification de la Surface de Bord. L'extraction de la surface de bord est réalisée en stockant les quatre faces de chaque tétraèdre lors du parcours du maillage. Dès qu'une paire de faces est trouvée, celle-ci est enlevée de la liste de stockage ce qui diminue drastiquement l'occupation mémoire comme illustré dans le tableau 2.5. À la fin de ce processus, ne restent que les faces singleton qui sont les faces de la surface de bord.

La surface est ensuite simplifiée en mémoire centrale. Toute méthode de simplification est applicable. Nous avons choisi d'utiliser deux d'entre elles se révélant être celles qui sont les plus implantées dans les logiciels de modélisation et de visualisation surfacique : effondrement d'arêtes guidé par une erreur métrique quadrique [GH97] ou décimation de sommets [SZL92][1].

Extraction des points critiques du champ scalaire. On définit les points critiques comme l'ensemble des sommets où le champ scalaire est un extremum local. On simplifie ainsi les points internes (c'est-à-dire n'appartenant pas au bord) du maillage tétraédrique en ne conservant que les points critiques. Bien que

[1] La décimation de sommets consiste à enlever à chaque étape l'étoilé d'un sommet et à retrianguler le trou généré.

	BuckyBall	Fighter (r)	Spx (r)	Sf1	Engine
Tétraèdres initiaux	1 250 235	5 929 085	10 911 011	13 980 162	41 943 040
Extraction du Bord	26	287	586	930	1192
Ratio	0.57%	3.13%	3.04%	0.13%	1.53%
Méthode	QEM	VC	VC	QEM	QEM
Temps total (en s)	35	306	625	1020	1507

TAB. 2.4: **Temps et mesures d'erreur de simplification de maillages tétraédriques**. Les temps sont en secondes. Le ratio du maillage simplifié par rapport au nombre de tétraèdres initiaux ainsi que la méthode utilisée : erreur quadrique (QEM) ou par clusterisation de sommets (VC) ; pour la simplification surfacique sont indiqués. Les maillages Fighter (r) et Spx (r) ont été générés en utilisant une CDT. Le maillage Engine est une grille régulière tétraédralisée.

cela ne garantit pas que ces points critiques le soient encore dans le maillage grossier final, en pratique la préservation topologique du champ scalaire est assurée comme l'illustre la figure 2.11. Notons que l'extraction supplémentaire des points selles est aussi envisageable comme proposée par [CL03].

L'extraction de ces points critiques est réalisée en comparant la valeur de chaque sommet avec les sommets de son lien ce qui assure la localité de la méthode. Aucun *Kd-tree* n'est donc utilisé pour échantillonner le champ scalaire ce qui ne limite pas la taille des maillages traités.

Création d'un maillage grossier en utilisant une tétraédralisation contrainte de Delaunay. La dernière étape consiste à générer un maillage tétraédrique contraint à la surface simplifiée et aux points critiques. Afin de garantir une certaine qualité de tétraèdres au sein du maillage grossier final, une CDT est réalisée via la bibliothèque *TetGen* [SG05].

2.2.3 Résultats

L'ensemble des résultats est répertorié au sein des tableaux 2.4 et 2.5. Nous nous intéressons à la fois au temps de simplification mais aussi à la différence entre le maillage fin initial et le maillage grossier extrait. Tous les tétraèdres générés ont un ratio rayon-arête (cf. définition 1.10) inférieur à deux ce qui garantit leur qualité [Si06]. Pour enlever les *slivers* possibles, une étape de raffinement de Delaunay est réalisée par l'algorithme de tétraédralisation.

Concernant les temps de simplification rassemblés dans le tableau 2.4, bien que notre solution soit pensée pour assurer une diminution de l'occupation mémoire (ce qui diminue les performances), ceux-ci sont équivalents à ceux de [UBF+05] pour les maillages inférieurs à un million et demi de tétraèdres, taille maximale référencée dans leur article. Avec nos améliorations, nous avons pu traiter plus de quarante millions de tétraèdres en environ 25 minutes (1507 secondes) pour le réduire à $1,53\%$ de la taille originale. Concernant l'ensemble de données *sf1* de près de quatorze millions de tétraèdres, son temps de simplification est de l'ordre de 17 minutes pour obtenir un maillage grossier représentant $0,13\%$ du maillage initial avec une garantie sur la qualité des tétraèdres. [VCL+07] réalise cette même simplification en 44 minutes (40 minutes de mise en forme du fichier $+$ 4 minutes de simplification proprement dite) pour obtenir un maillage réduit à 10% sans garantie sur la qualité des tétraèdres.

Afin d'éprouver la conservation à la fois du volume et du champ scalaire, nous avons aussi mesuré un certain nombre d'erreurs relatives aux surfaces de bord et au champ scalaire qui sont regroupées au sein du tableau 2.5. Pour calculer la distance de Hausdorff d_H entre la surface fine initiale et la surface grossière, l'outil *Metro*[2] de Cignoni a été utilisé. Pour mesurer les écarts au sein du champ scalaire, nous avons utilisé l'outil *TetMesh Comparator*[3] de Bavoil. Dues aux limitations d'implantation de ces outils de mesure d'erreurs, nous ne pouvons fournir des résultats pour les maillages dont le nombre de tétraèdres dépasse six millions.

On peut noter que dans leur ensemble, la différence entre les deux maillages concernant le volume occupé est faible au sens de la distance de Haussdorf. Afin de mesurer la différence entre deux champs scalaires,

[2] *Metro* est disponible à l'adresse suivante : http ://vcg.isti.cnr.it/activities/surfacegrevis/simplification/metro.html.
[3] *TetMesh Comparator* est disponible à l'adresse suivante : http ://ww.sci.utah.edu/b̃avoil/research/tetsimp/tmc/.

	Comb	BuckyBall	Fighter (r)
Tétraèdres initiaux	215 040	1 250 235	5 929 085
Erreurs			
Distance de Hausdorff d_H	0,131810	0,000019	0,000068
Intervalle Scalaire	$[0,197813; 0,710419]$	$[0; 254]$	$[0; 2,9]$
Erreur maximum ε_{max}	0,391	250	0,858
Erreur moyenne ε_{min}	0,0135	56	0,0578
Ecart-type σ	0,0220	71,8	0,0753
Moyenne Quadratique $\bar{\varepsilon}$	0,0258	91,5	0,0949
Espace Mémoire			
Extraction de la surface de bord *in-core*	1136 Ko	57804 Ko	376 Mo
Notre technique	1212 Ko	4404 Ko	190 Mo

TAB. 2.5: **Mesures d'erreurs et mémoire de la simplification de maillages tétraédriques**. Les erreurs entre le maillage d'origine et le maillage généré sont indiquées. Les erreurs maximum ε_{max} , moyenne ε_{min}, écart type σ et moyenne quadratique $\bar{\varepsilon}$ concernent le champ scalaire, la distance de Hausdorff d_H mesure quant à elle la différence entre les deux surfaces de bord. Enfin, la différence d'occupation mémoire entre une extraction *classique* de la surface de bord et notre technique est fournie.

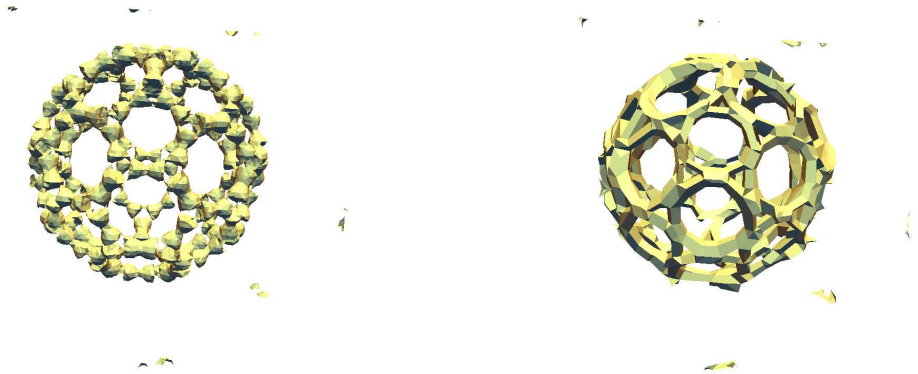


FIG. 2.11: **Conservation de la topologie du champ scalaire**. À gauche, une isosurface extraite du maillage *BuckyBall* à haute résolution (1 250 235 tétraèdres). La même isosurface à droite extraite de notre maillage simplifié (7 126 tétraèdres) en n'utilisant que les extrema locaux (hors points selles).

TetMesh Comparator échantillonne ceux-ci à la fois sur les sommets mais aussi en prenant des points aléatoires au sein des tétraèdres. En faisant tendre ce nombre de points vers l'infini, on peut ainsi vérifier l'égalité continue des deux champs. Néanmoins en pratique, nous avons pris cinq points au sein de chaque tétraèdre afin de respecter le protocole expérimental réalisé dans [UBF+05]. Les erreurs maximales ε_{max} et moyenne ε_{min} ainsi que l'écart-type σ et la moyenne quadratique $\bar{\varepsilon}$ sont calculés. On notera qu'excepté pour l'ensemble de données *BuckyBall*, l'erreur moyenne et l'écart type sont faibles ce qui corrobore le fait que l'extraction des points critiques est suffisante pour reconstruire un champ scalaire proche de l'original. Pour l'ensemble *BuckyBall*, les résultats sont plus critiquables et pourraient remettre en cause notre précédente conclusion. Mais en effectuant plusieurs extractions d'isosurfaces, comme celle réalisée au sein de la figure 2.11, on note que malgré une erreur moyenne non négligeable la topologie du champ scalaire est préservée et donc que le champ scalaire grossier conserve les propriétés remarquables du champ d'origine.

2.3 Partition d'un maillage

Afin de pouvoir créer un maillage birésolution à partir des maillages fin et grossier, une correspondance entre les deux résolutions doit être établie. En effet, lors de l'exploration des données, une partie du maillage grossier va être remplacée par une partie du maillage fin *via* la définition d'erreurs d'approximation dynamiques. Or, pour réaliser cet ajout d'informations, on doit savoir précisément quels simplexes fins remplaceront les simplexes grossiers. Cela ne peut se faire qu'en ayant une relation précise entre les deux niveaux de détails

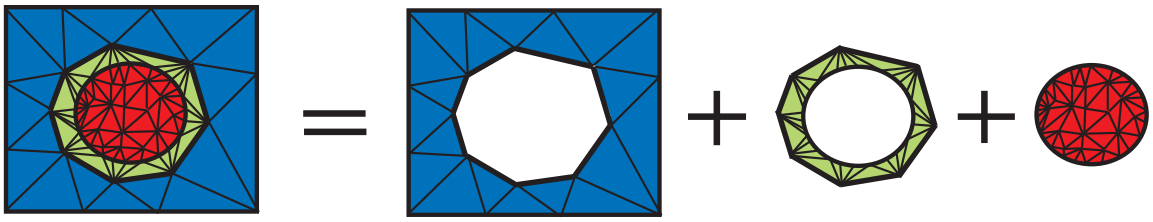


FIG. 2.12: **Maillage Birésolution** : Le maillage birésolution est composé de tétraèdres grossiers (en bleu), de tétraèdres fins (en rouge) et de tétraèdres *de lien* assurant la jonction (en vert).

à la manière des graphes de dépendances pour les schémas multirésolution.

Pour cela, nous définissons une surjection entre les sommets des deux différentes résolutions. Cette surjection est corrélée au calcul d'une partition spatiale des sommets fins guidée par la répartition spatiale des sommets grossiers.

Nous exprimons dans un premier les objectifs que doit vérifier la partition dans le cadre du maillage birésolution (section 2.3.1) ; puis nous détaillons la problématique plus générale dans laquelle elle s'incrit grâce à un bref état de l'art (section 2.3.2). Nous introduisons la notion de validité topologique d'une partition au sein de la suivante section 2.4. Les exemples concrets de partition seront évoqués au sein des sections 2.5 et 2.6 suivantes. De plus, pour des raisons de clarté, l'ensemble des figures dans la suite de ce chapitre sont représentées sur des maillages triangulaires sans perte de généralité.

2.3.1 Objectifs de la partition

La définition d'une partition du maillage fin contrainte par le maillage grossier doit permettre la création d'un maillage birésolution combinant une partie de ces deux maillages. Cela veut dire, comme illustré au sein de la figure 2.12, que le maillage birésolution est composé de trois ensembles de tétraèdres distincts : des tétraèdres appartenant au maillage grossier (en bleu), des tétraèdres appartenant au maillage fin (en rouge) et des tétraèdres dit *de lien* assurant la jonction entre les deux résolutions (en vert). La partition conditionne le maillage birésolution avec l'hypothèse que si un sommet grossier est conservé alors ses sommets fins associés ne le seront pas (et réciproquement).

Pour cela, la partition doit vérifier un ensemble de conditions afin d'assurer que le maillage final soit *conforme*. Un maillage *conforme* est un complexe simplicial. Il n'a donc aucune intersection géométrique entre ses cellules. Il est valide à la fois topologiquement et géométriquement. Les conditions sont les suivantes :

C.i. La partition permet d'obtenir le maillage grossier à partir du maillage fin.

C.ii. Il n'y a pas d'intersection entre les tétraèdres fins et les tétraèdres grossiers composant le maillage birésolution.

C.iii. Il n'y a pas d'intersection entre les différents tétraèdres de lien et avec les tétraèdres fins comme les tétraèdres grossiers conservés composant le maillage birésolution.

La condition *C.i* assure la validité topologique du maillage birésolution alors que les suivantes *C.ii* et *C.iii* la validité géométrique. La vérification de ces trois conditions garantit ainsi un maillage conforme.

2.3.2 Méthodes de partitionnement existantes

Notre problématique d'extraction d'une partition spatiale des sommets du maillage fin entre dans le cadre de problèmes plus généraux de partitionnement de maillages et de graphes que nous allons évoquer brièvement au sein de cette partie. En effet, le *regroupement* des simplexes d'un maillage – on parle aussi de *clusterisation* ou de *partitionnement* ou encore de *segmentation* en fonction des applications ; peut intervenir dans de nombreux domaines comme l'animation, le *morphing*, la sémantique ou le parallélisme. Le lecteur intéressé par ce sujet

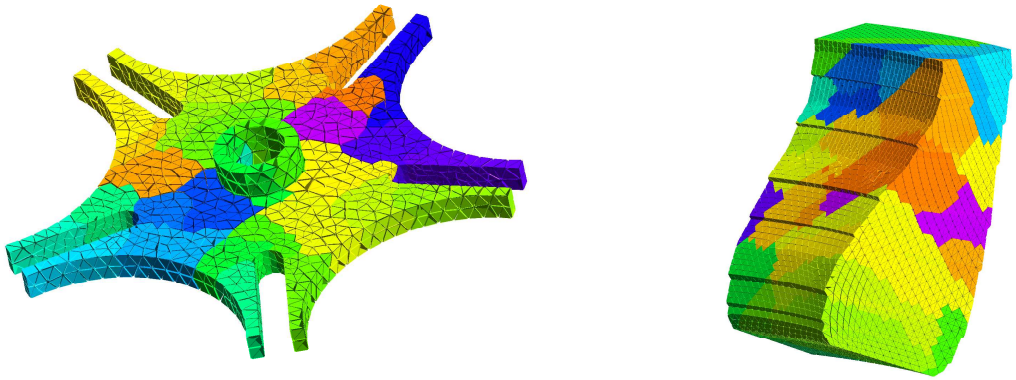


FIG. 2.13: ***K-way Partitioning* de maillages tétraédriques**. Regroupement de sommets pour des maillages tétraédriques en $k = 30$ sous-ensembles indépendants en utilisant le logiciel *Metis* [KK98]. À gauche, une pièce mécanique. À droite, une chambre à combustion (*Comb*). Cet algorithme n'est pas applicable à notre cas, car la partition de la tétraédralisation fine doit dépendre du maillage grossier.

peut se référer par exemple à l'état de l'art proposé par Ariel Shamir pour les maillages triangulés [Sha08]. Notons que l'approche hiérarchique de segments proposée par [SS06] précédemment détaillée en section 1.2 entre dans cette catégorie.

La problématique abordée lors de ces partitionnements peut se rapprocher d'un unique problème de minimisation façonné en fonction du contexte [Sha08] :

Problème 2.1 Regroupement de maillages comme un problème d'optimisation. *Soit un maillage Σ et S l'ensemble des d-simplexes de Σ. La problématique de trouver un partitionnement disjoint de S en $S_0, \ldots, S_{k-1}$ peut être considéré comme un problème d'optimisation portant sur une fonction $J = J(S_0, \ldots, S_{k-1})$ dépendante de $S_0, \ldots, S_{k-1}$ telle que J soit minimisée (ou maximisée) sous un certain nombre de contraintes C.*

Néanmoins, l'ensemble des contraintes C vérifiées lors du partitionnement ne concerne qu'un seul maillage. En effet, celles-ci ciblent le nombre de sous-ensembles S_k, leur taille, leur convexité, ou encore leur topologie. Mais les contraintes ne sont jamais liées à un autre sous-ensemble d'éléments indépendants à Σ comme cela est le cas dans la formulation de notre problème, où elles sont dépendantes du maillage grossier.

Ainsi, des solutions devenues des "standards" – comme le *k-way partitioning* de Karypis et Kumar [KK98][4] utilisé à la fois pour les graphes et les maillages afin de créer des sous-ensembles équilibrés en minimisant les arêtes connectant les sommets appartenant à différents sous-ensembles – ne peuvent être utilisées au sein de notre approche. Une illustration de la partition obtenue sur deux maillages tétraédriques pour $k = 30$ est disponible en figure 2.13.

2.4 Validité topologique de la partition

Dans cette section, on définit une partition des sommets d'un maillage $\Sigma = (V, T)$ où V est l'ensemble des sommets et T l'ensemble des tétraèdres. On pose $\Sigma_f = (V_f, T_f)$ et $\Sigma_c = (V_c, T_c)$ une même simulation à deux granularités différentes telles que respectivement Σ_f soit la résolution fine et Σ_c la résolution grossière.

Une partition des sommets d'un maillage Σ peut être définie de la manière suivante :

Définition 2.2 Partition des sommets d'un maillage : On dit que $(S_k)_k$ est une partition des sommets de Σ si :

i. $(S_k)_k$ sont des sous-ensembles de V.

ii. $\sqcup S_k = V$.

Dans ce cas, les sous-ensembles S_k sont appelés des **clusters**.

[4] [KK98] est l'un des articles fondateurs du logiciel *Metis* permettant de réaliser rapidement des regroupements au sein de graphes et de maillages, disponible à l'adresse suivante : http ://glaros.dtc.umn.edu/gkhome/views/metis.

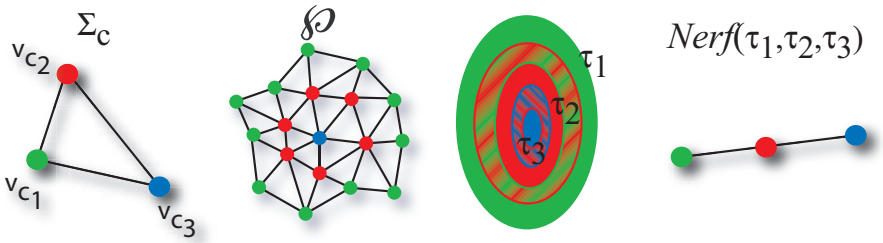


FIG. 2.14: **Premier exemple de partition n'assurant pas la reconstruction du maillage grossier**. Le maillage grossier Σ_c est un triangle à gauche composé des sommets v_{c_1}, v_{c_2} et v_{c_3}. La partition des sommets fins extraite de $\wp$ est donnée au centre, chaque sommet fin étant coloré en fonction de son image par la surjection $\wp$ sur le maillage fin. La partition abstraite vu comme les ensembles τ_1, τ_2 et τ_3 est aussi dessinée. Il n'existe aucun triangle fin dont les trois sommets ont une couleur différente ce qui correspond à l'absence d'intersection entre les trois clusters τ_i. On constate à droite que le nerf $Nerf(\bar{\wp}(T_f)) = Nerf(\tau_1, \tau_2, \tau_3)$ est différent de Σ_c. $\wp$ ne vérifie donc pas $C.i$.

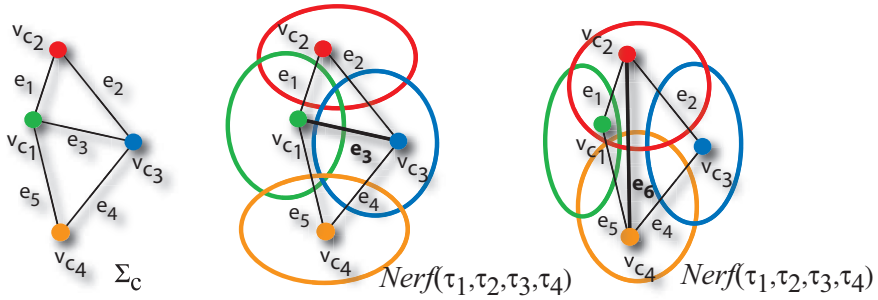


FIG. 2.15: **Second exemple de partition n'assurant pas la reconstruction du maillage grossier**. Le maillage grossier Σ_c est composé de deux triangles à gauche. Basé sur celui-ci deux partitions peuvent être extraites au niveau du maillage fin (représentées de manière abstraite). L'une d'entre elles reconstruit Σ_c, l'autre non puisque l'arête e_3 est remplacé par l'arête e_6. $\wp$ ne vérifie donc pas $C.i$.

Afin de réaliser une partition de notre maillage fin Σ_f contrainte par le maillage grossier Σ_c, nous définissons en premier lieu une surjection $\wp$ des sommets fins vers les sommets grossiers formulée de la manière suivante :

$$\begin{array}{rccc} \wp : & V_f & \rightarrow & V_c \\ & v_f & \rightarrow & c = \wp(v_f) \end{array}$$

On définit l'ensemble $S_c \subset V_f$ comme l'image inverse d'un sommet grossier c par $\wp$:

$$S_c = \{v_f \in V_f / \wp(v_f) = c\}$$

On étend $\wp$ aux tétraèdres fins de la manière suivante : chaque tétraèdre fin est associé à un ensemble de sommets grossiers qui sont les images de ses sommets fins par $\wp$. Ainsi, un tétraèdre fin est associé à un, deux, trois voire quatre sommets grossiers. On définit donc $\bar{\wp} : T_f \rightarrow \mathbb{P}_c$ où $\mathbb{P}_c$ est l'ensemble des parties de V_c par :

$$\forall\, t_f = (v_{f_i})_{i \in [0..3]} \in T_f,\ \bar{\wp}(t_f) = \{c \in V_c / \wp(v_{f_i}) = c,\ i \in [0..3]\}$$

On définit de la même manière l'ensemble τ_c comme l'ensemble des tétraèdres fins $t_f \in T_f$ dont au moins un des sommets a pour image le sommet grossier c par $\bar{\wp}$:

$$\tau_c = \{t_f \in T_f / c \in \bar{\wp}(t_f)\} \tag{2.1}$$

Afin de vérifier la première condition $C.i$, la fonction $\wp$ permettant de caractériser la partition doit vérifier la condition suivante :

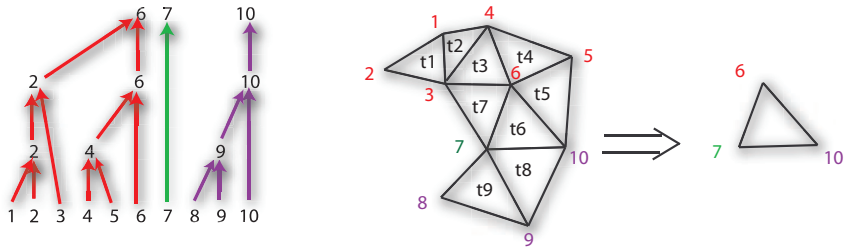


FIG. 2.16: **Partition issue d'un processus de simplification**. À gauche, graphe d'exécution des contractions d'arêtes. Les feuilles sont les sommets fins, les racines les sommets grossiers. Tous les sommets fins se retrouvant projetés sur le même sommet grossier appartiennent au même cluster. À droite, les maillages triangulés fin et grossier associés.

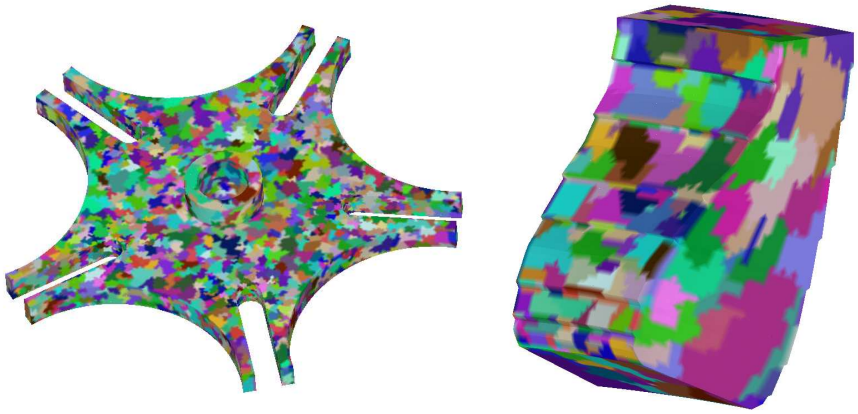


FIG. 2.17: **Exemples de partitions issues d'un processus de simplification basé sur [Viv05]**. À gauche, 14 139 clusters pour le maillage *Piece*. À droite, 1 030 clusters pour le maillage *Comb*.

Condition 2.1 Validité topologique de la surjection $\wp$. *Soient $\wp$ une surjection entre les sommets fins et les sommets grossiers et $\tau_1, \ldots, \tau_{card\ V_c}$ les ensembles de tétraèdres fins définis par (2.1) pour chacun des sommets grossiers de V_c. On dit que $\wp$ est topologiquement valide si et seulement si :*

$$Nerf(\tau_1, \ldots, \tau_{card\ V_c}) = \Sigma_c \tag{2.2}$$

$\wp$ vérifie alors la condition $C.i$. En effet, si on dispose de $\wp$, les $\tau_1, \ldots, \tau_{card\ V_c}$ peuvent être déterminés.

Si cette condition n'est pas vérifiée, plusieurs situations peuvent arriver :

– certains simplexes de Σ_c peuvent ne pas être reconstruits comme illustré au sein de la figure 2.14. Le maillage grossier Σ_c est un triangle composé des sommets v_{c_1}, v_{c_2} et v_{c_3}. La partition des sommets fins extraite de $\wp$ ne génère aucun triangle fin dont les trois sommets ont une couleur différente Cela provoque l'absence d'intersection entre les trois clusters τ_i correspondant. Le nerf $Nerf(\tau_1, \tau_2, \tau_3)$ issu de $\overline{\wp}$ est ainsi différent de Σ_c. $\wp$ ne vérifie donc pas $C.i$;
– des basculements d'arêtes ou de faces peuvent se produire comme illustré au sein de la figure 2.15. Le maillage grossier Σ_c est composé de deux triangles. Deux partitions peuvent être extraites au niveau du maillage fin. Néanmoins, l'une d'entre elles reconstruit Σ_c mais pas l'autre car une arête (l'arête e_3) est remplacé par une autre arête (l'arête e_6). Les deux maillages diffèrent en effet d'un basculement d'arêtes. $\wp$ ne vérifie donc pas $C.i$.

Afin de vérifier les conditions $C.ii$ et $C.iii$, $\wp$ ne doit pas être uniquement valide topologiquement. Elle doit aussi vérifier des contraintes géométriques qui seront développées au sein de la section 2.6.

2.5 Partition issue d'un processus de simplification

2.5.1 Définition

La première partition proposée repose directement sur un processus de simplification basé sur des effondrements de demi-arêtes (*half-edge collapses*). Le maillage grossier utilisé est ainsi le maillage issu du processus

	Comb	Piece	BuckyBall	Engine70
Tétraèdres initiaux	245 040	889 157	1 250 235	5 152 961
Tétraèdres finaux	5 305	77 828	28 474	441 979
Ratio simplification	2,16 %	8,75%	2,28%	8,58%
Temps (en secondes)	42	257	263	1208

TAB. 2.6: **Temps de calcul pour extraire la partition du processus de simplification** *via* l'algorithme de [Viv05]. Le nombre de tétraèdres initial et final est indiqué. Le temps est en secondes. L'ensemble *Engine70* est une extraction tétraédrique d'une partie de la grille régulière du même nom correspondant à l'isosurface de valeur 70.

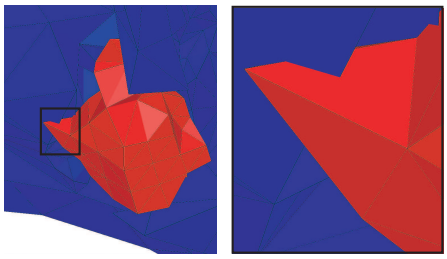


FIG. 2.18: **Limitation de l'approche reposant sur un algorithme de simplification**. Situation d'extraction dynamique avec un zoom sur la zone à problème (à droite). Le maillage grossier est affiché en bleu, les clusters extraits en rouge. On remarque qu'il y a une intersection non vide entre les deux maillages ce qui entraînera lors de l'extraction la création d'un maillage non conforme pouvant généré des artefacts lors de la visualisation.

de simplification et n'est donc pas arbitraire.

La simplification par demi-arêtes repose sur l'élimination d'un sommet en l'effondrant sur un autre (comme décrit en section 1.1.1). Cela permet de définir implicitement une partition du maillage fin comme l'illustre la figure 2.16. On dira que les sommets (u,v) appartiennent au même cluster S_v si l'arête $e=(u,v)$ est effondrée de u vers v.

Nous avons implanté la construction de cette partition au sein d'un algorithme de simplification [Viv05]. L'algorithme permettant de l'extraire est donné ci-après :

***ALGORITHME* 1**

```
Initialisation : Pour tout sommet v ∈ V_f, initialiser S_v à {v}.
Tant qu'une arête e = (v,w) est contractée
    Si v → w, S_w = S_w ∪ S_v
    Sinon S_v = S_v ∪ S_w
```

L'algorithme précédent permet la création d'une partition valide topologiquement (vérifiant la condition $C.i$). En effet, d'après [Ede01], pages 81-82, une suite de contractions locales d'arêtes conservant la topologie permet de garantir un isomorphisme entre le nerf issu des étoilés de $S_c = \wp^{-1}(c)$ et Σ_c. Or pour chaque sommet grossier c, on a $Etoile\ S_c = \tau_c$ puisque par définition de l'étoilé (cf. définition 1.3) , $Etoile\ S_c$ contient tous les tétraèdres fins ayant au moins un sommet dont l'image par $\wp$ est c. L'égalité est obtenue grâce à l'utilisation des sommets grossiers V_c pour former le maillage grossier résultant du nerf. On obtient ainsi la condition 2.1.

2.5.2 Résultats et Limitations

Complexité Temporelle Cette première solution reposant sur un algorithme de simplification par effondrements de demi-arêtes permet d'obtenir directement la partition en sauvegardant l'historique des opérations locales – comme illustrée en figure 2.16. Elle permet ainsi de coupler la simplification et l'extraction de la partition en une unique étape. Les résultats obtenus pour deux ensembles de données sont présentés en figure 2.17. Les temps d'extraction de la partition sont alors équivalents à ceux de la simplification et sont renseignés au sein du tableau 2.6. La simplification d'un maillage de plus de cinq millions de tétraèdres nécessite tout de même vingt minutes. Pour comparaison, notre algorithme de simplification (détaillé en section 2.2) réalise cette opération en trente secondes (contre quatre minutes) pour l'ensemble *BuckyBall* de plus d'un million de tétraèdres et en cinq minutes pour un maillage ayant presque six millions de tétraèdres.

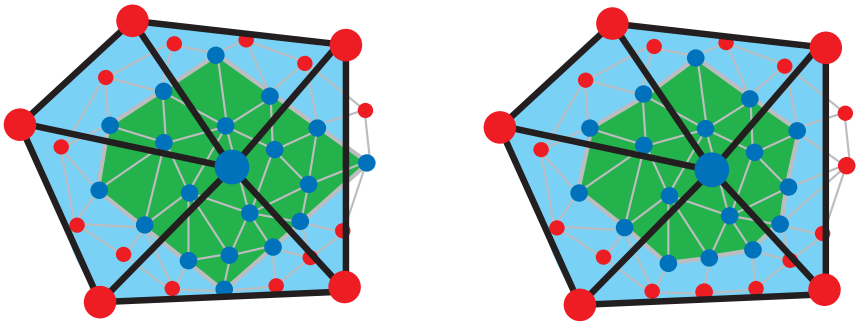


FIG. 2.19: **Validation géométriquement faible**. Le maillage fin est représenté avec des arêtes grises avec la partition de ses sommets fins (bleu ou rouge). Le maillage grossier est représenté avec des arêtes noires. Le sommet c est le sommet bleu. L'étoilé *Etoile c* de c est en bleu clair, l'ensemble $\overline{\tau}_c$ en vert. À gauche, $\overline{\tau}_c$ n'est pas inclus dans *Etoile c*, la condition 2.2 n'est pas vérifiée. À droite, celle-ci est vérifiée : il n'y a pas d'intersections entre les éléments fins et grossiers dans le maillage birésolution.

Limitations Néanmoins, cette solution a plusieurs inconvénients. Tout d'abord, cette solution va à l'encontre des prérequis que nous avons imposés en proposant notre algorithme de simplification en section 2.2. En effet, la simplification par décimation locale repose sur la construction de structures de données coûteuses en place mémoire qui empêchent le traitement de gros maillages de données. Ainsi l'algorithme ne peut pas traiter des maillages dont la taille est supérieure à six millions de tétraèdres (pour une mémoire vive de deux mégaoctets).

D'un point de vue pratique et utile, il est important de rappeler que la construction de deux résolutions d'une même simulation sont souvent indépendantes. Il est alors dans ce cas précis impossible de construire une partition entre les deux résolutions en se basant sur un algorithme de simplification. De plus, cette solution impose de pouvoir modifier les algorithmes de simplification afin d'y insérer la sauvegarde de la construction de la partition. Au sein de logiciels de visualisation commerciaux, cette solution est rarement implantable directement. Cette solution semble donc peu enclin à être efficace à l'utilisateur final.

De plus, cette technique n'apporte qu'une seule garantie géométrique lors de l'extraction dynamique : celle que le sommet grossier est inclus au sein de son cluster associé. Ce qui n'est pas suffisant pour vérifier les conditions *C.ii* et *C.iii* définies en section 2.3.1. En effet, comme illustré au sein de la figure 2.18, cette solution peut aboutir à des intersections entre le maillage fin (en rouge) et le maillage grossier (en bleu) qui sont conservés lors de la construction de notre maillage birésolution. Une telle solution ne garantit donc nullement la condition *C.ii*.

2.6 Partition issue de critères géométriques

Afin de répondre aux problématiques soulevées lors de l'évaluation de notre première partition, nous proposons une seconde partition indépendante du processus de simplification et plus précisément des effondrements de simplexes. Elle repose uniquement sur des critères géométriques et peut être ainsi appliquée dans n'importe quel contexte : maillage grossier indépendant ou obtenu via l'algorithme de simplification proposé en section 2.2.

2.6.1 Critère de libre intersection entre le maillage grossier et le maillage fin

Condition 2.2 Validité géométrique faible de la partition $\wp$. *Soit c un sommet grossier de Σ_c et $\overline{\tau}_c = \overline{\wp}^{-1}(c)$ l'ensemble des tétraèdres fins dont tous les sommets ont pour image c par $\wp$. On dit que $\wp$ est faiblement valide géométriquement si et seulement si :*

$$|\overline{\tau}_c| \subset |Etoile\ (c)| \tag{2.3}$$

où $|\overline{\tau}_c|$ (respectivement $|Etoile\ (c)|$) est l'espace occupé par $\overline{\tau}_c$ (respectivement $Etoile\ (c)$).

La figure 2.19 illustre cette condition. L'ensemble $\overline{\tau}_c$ est coloré en vert. La partition de gauche ne la vérifie pas alors que la droite la vérifie.

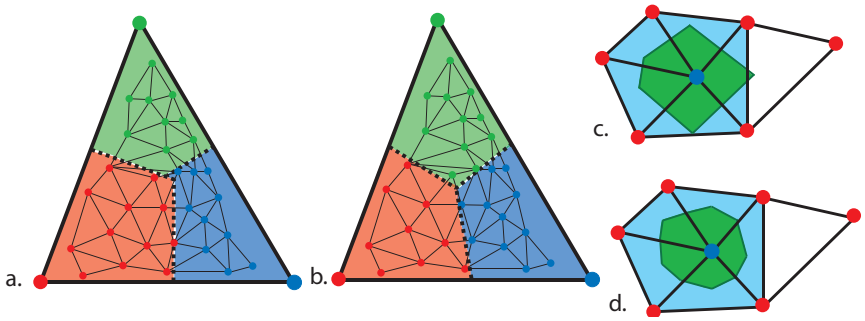


FIG. 2.20: **Comparaisons entre le diagramme de Voronoï et les coordonnées barycentriques**. (a-b.) Le maillage fin triangulaire est dessiné au sein du triangle grossier dans lequel on applique la partition selon Voronoï ou les coordonnées barycentriques. La limite géométrique séparant les clusters est représentée par des lignes en pointillés. La partition obtenue selon les sommets grossiers contient trois clusters bleu, rouge et vert (en accord avec la coloration des dits sommets). (a.) Exemple d'une partition de Voronoï. (b.) Exemple d'une partition basée sur les coordonnées barycentriques. (c-d.) Le maillage représente la résolution grossière, la fine est omise. On s'intéresse au sommet grossier bleu c. Son étoilé $Etoile\ c$ est en bleu clair. $\overline{\tau}_c$ est en vert. (c.) Exemple de violation de la condition (2.2) en utilisant le diagramme de Voronoï. $\overline{\tau}_c \not\subset Etoile\ c$, puisque une partie de $\overline{\tau}_c$ intersecte aussi le triangle blanc ne faisant pas partie de $Etoile\ c$. (d.) En utilisant les coordonnées barycentriques, le problème précédent n'est plus possible, *via* leur définition, la condition (2.2) est vérifiée.

2.6.2 Basée sur un diagramme de Voronoï

Une idée intuitive pour extraire une telle partition est d'utiliser le diagramme de Voronoï [Aur91] associé aux sommets grossiers V_c. En effet, comme vu en définition 1.14, le *nerf* du diagramme de Voronoï n'est autre qu'une tétraédralisation de Delaunay. Une telle partition est présentée pour un maillage fin donné au sein de la figure 2.20 (a).

Définition 2.3 Surjection définie par le diagramme de Voronoï $\wp_{\mathcal{V}}$:

$$\wp_{\mathcal{V}}(v_f) = \min_{v_c \in V_c} \left\{ d(v_f, v_c) / v_c \in V_c \right\}$$

Néanmoins, cette définition ne permet pas de vérifier la condition (2.2). Un contre-exemple à cette condition est fournie au sein de la figure 2.20 (c.). Ainsi, l'inclusion du cluster au sein de l'étoilé n'est aucunement garantie par $\wp_{\mathcal{V}}$.

2.6.3 Basée sur les coordonnées barycentriques

Pour essayer de répondre aux problématiques soulevées par la précédente définition, nous avons revu les contraintes géométriques afin de vérifier la condition 2.2.

Pour cela, nous utilisons les coordonnées barycentriques au sein des tétraèdres grossiers pour assigner à chaque sommet fin son cluster d'appartenance. La nouvelle définition pour $\wp$ est donc la suivante :

Définition 2.4 Surjection définie sur les coordonnées barycentriques $\wp_{\mathcal{B}}$:

$$\wp_{\mathcal{B}}(v_f) = \max_{v_c \in V_c} \left\{ \alpha_{i_c} / v_f = \sum_{i_c \in [1..card\ V_c]} \alpha_{i_c} v_c \right\}$$

avec $\sum \alpha_{i_c} = 1$ en posant $\alpha_{i_c} = 0$ si v_c n'appartient pas au tétraèdre grossier contenant v_f.

Cela veut dire qu'on associe à chaque sommet fin v_f, son sommet grossier le plus proche c dans l'espace barycentrique défini au sein du tétraèdre grossier auquel il appartient. Un exemple de la partition pouvant être obtenue grâce à cette définition est proposé en figure 2.20 (b).

Néanmoins, $\wp_{\mathcal{B}}$ n'est définie que dans l'espace occupé par le maillage grossier Σ_c (noté $|\Sigma_c|$). Or, certains sommets fins de V_f peuvent se trouver en dehors de cet espace. Cela peut être dû à une mauvaise conservation

	BuckyBall	Fighter (r)	Spx (r)	Sf1	Engine
Tétraèdres initiaux	1 250 235	5 929 085	10 911 011	13 980 162	41 943 040
Temps (en s)	28	360	1518	990	1230

TAB. 2.7: **Temps d'extraction de la partition basée sur les coordonnées barycentriques :** Les temps sont en secondes. Ils sont à associer avec les temps du tableau 2.4. Cette méthode permet de traiter de plus gros maillages que la première reposant sur un algorithme glouton de simplification.

du volume – soit en terme d'erreur, à une distance de Hausdorff importante entre les surfaces de bord des deux résolutions. On étend ainsi $\wp_{\mathcal{B}}$ pour ces points en utilisant $\wp_{\mathcal{V}}$. On a ainsi la définition finale suivante pour $\wp$:

Définition 2.5 Surjection étendue $\wp$:

$$\wp(v_f) = \begin{cases} \wp_{\mathcal{B}}(v_f) \ si \ v_f \in |\Sigma_c| \\ \wp_{\mathcal{V}}(v_f) \ sinon \end{cases}$$

où $|\Sigma_c|$ est par définition l'espace occupé par le maillage Σ_c.

L'utilisation des coordonnées barycentriques garantit par définition que la condition (2.2) est vérifiée pour chaque ensemble de tétraèdres fins $\overline{\tau}_c$. La figure 2.20 (d) illustre la modification obtenue pour $\overline{\tau}_c$ en remplaçant le diagramme de Voronoï par les coordonnées barycentriques.

2.6.4 Résultats et Limitations

Au sein de cette section, nous présentons tout d'abord les deux avantages principaux de la surjection $\wp$ définie grâce à l'utilisation des coordonnées barycentriques : l'indépendance du maillage grossier et la faible complexité temporelle et mémorielle. Puis, nous évoquons la validité topologique de $\wp$. Enfin, nous discutons des limitations liées à une telle solution et leurs conséquences sur l'extraction d'un maillage birésolution.

Indépendance du maillage grossier Tout d'abord, une définition pour $\wp$ reposant uniquement sur des critères géométriques permet d'utiliser un maillage grossier associé au maillage initial ne dépendant pas d'un processus de simplification particulier. Il peut même être généré en tant que discrétisation de l'espace par un logiciel de création de maillages.

Néanmoins, en pratique, pour assurer le bon déroulement du calcul de $\wp$, le maillage grossier doit vérifier un certain nombre de critères :

- L'espace occupé par le maillage fin Σ_f doit avoir une intersection non vide avec l'espace occupé par le maillage grossier Σ_c ($|\Sigma_f| \cap |\Sigma_c| \neq \emptyset$) afin que l'utilisation des coordonnées barycentriques ait un sens. Cela exclut les situations où les deux maillages se trouvent dans une région de l'espace complètement différente. Établir une correspondance entre ces deux maillages n'aurait d'ailleurs aucun sens dans l'optique finale de reconstruire un maillage birésolution.
- Pour assurer la reconstruction de chaque tétraèdre grossier, ceux-ci doivent **tous** contenir au moins un tétraèdre fin **différent** (dont l'image par $\wp$ sera le tétraèdre grossier). Cette restriction impose que la granularité du maillage grossier soit inférieure globalement à celle du maillage fin et que l'espace occupé par le maillage grossier est inclus strictement dans l'espace occupé par le maillage fin ($|\Sigma_c| \subset |\Sigma_f|$).
- Le maillage grossier ne doit pas contenir de tétraèdres dégénérés (comme des *slivers*) dont le déterminant est proche de zéro. Ils entraînent des instabilités numériques lors du calcul des coordonnées barycentriques, faisant échouer localement le calcul de $\wp$.

Le respect de ces trois critères assurent, en pratique, le bon déroulement du calcul de $\wp$. L'ensemble des tests effectués par la suite sont réalisés sur des maillages grossiers vérifiant ces critères.

En pratique, les maillages tétraédriques obtenus *via* des tétraédralisations de Delaunay [She98] ou des tétraédralisations contraintes de Delaunay [SG05] sont de bons candidats pour le bon déroulement du calcul de $\wp$. Les algorithmes de simplification conservant la qualité des tétraèdres grossiers comme celui proposé au sein de la section 2.2.2 mais aussi [CDM04, UBF+05] le sont aussi.

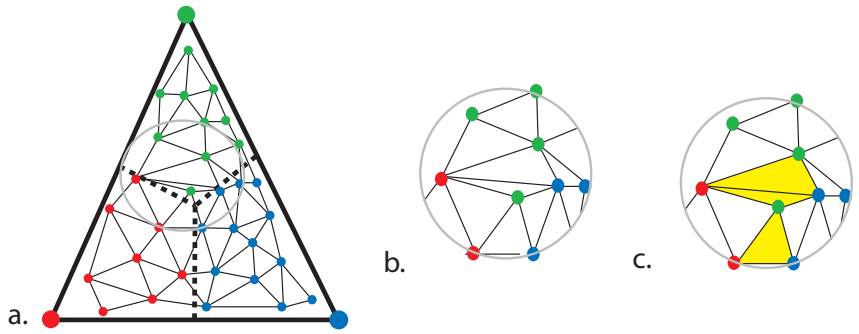


FIG. 2.21: **Problèmes pouvant être responsables de la non conformité de conditions pour les coordonnées barycentriques**. (a.) Exemple d'une partition obtenue en utilisant les coordonnées barycentriques. Le maillage fin triangulaire est dessiné au sein du triangle grossier dans lequel on applique la partition $\wp$. La limite géométrique séparant les clusters est représentée par des lignes en pointillés. La partition obtenue selon les sommets grossiers contient trois clusters bleu, rouge et vert (en accord avec la coloration des dits sommets). Un zoom est réalisé en (b.) et (c.) au sein du cercle gris. (b.) Le cluster vert n'est pas connexe. (c.) L'unicité entre un triangle fin et un triangle grossier n'est pas garantie. Ici, il existe trois triangles (en jaune) ayant trois couleurs différentes.

Complexité temporelle et mémorielle Le calcul des coordonnées barycentriques a un faible coût temporel. En effet, dans le pire cas, celle-ci est de l'ordre de $O(|V_f| \times |T_c|)$. Néanmoins, afin d'accélérer l'extraction de $\wp$ dans les situations où $|T_c|$ n'est plus négligeable, nous effectuons un premier découpage de l'espace où les tétraèdres du maillage grossier sont regroupés par boîte englobante *via* l'utilisation d'un *octree*. Pour chaque sommet fin v_f, la recherche n'est effectuée alors qu'au sein de la feuille de l'octree à laquelle il appartient. En pratique, cette implantation permet de réduite la complexité du calcul à $O(|V_f|)$.

Comme un traitement spécifique est réalisé pour les sommets se trouvant hors de l'espace occupé par le maillage grossier, un octree est aussi construit sur les sommets grossiers V_c afin d'accélérer la recherche du sommet grossier le plus proche.

Les temps de calcul sont renseignés au sein du tableau 2.7. Cette solution permet ainsi de traiter des maillages composés jusqu'à plus de quarante millions de tétraèdres en moins de 21 minutes. La combinaison de cette partition avec notre algorithme de simplification donne des temps de précalculs jusque là inégalés pour de telles masses de données. L'ensemble des valeurs sont fournies au sein de l'annexe B. Ainsi, notre algorithme de simplification avec notre algorithme de partitionnement réalisés l'un à la suite de l'autre indépendamment [5] permet de traiter un million de tétraèdres en une minute (contre quatre pour la solution reposant pour l'algorithme de simplification) et moins de six millions de tétraèdres en onze minutes (contre vingt minutes pour un peu plus de cinq millions de tétraèdres).

Cette définition de $\wp$ garantit aussi un faible coût mémoire. En effet, la partition ne reposant uniquement que sur le plongement des sommets fins au sein des tétraèdres grossiers, son calcul ne nécessite pas d'information de connectivité entre cellules, contrairement à la précédente solution (détaillée en section 2.5). La globalité des précalculs peuvent ainsi se réaliser sans le calcul et le stockage des relations topologiques comme pour notre algorithme de simplification.

Une telle approche garantit donc une faible complexité temporelle sans surcoût mémoire dû à des relations topologiques.

Surjection Topologiquement Valide Afin que $\wp$ soit topologiquement valide, celle-ci doit vérifier la condition (2.1). Cela veut dire que $\wp$ permet grâce à l'extraction des ensembles τ_c de tétraèdres fins, la reconstruction du maillage grossier initial Σ_c. Une idée intuitive pour affirmer que $\wp$ est topologiquement valide repose sur le fait que $\wp$ vérifie la condition géométrique faible. Dans ce cas précis, nous pouvons garantir que les τ_v associés aux sommets grossiers d'un tétraèdre grossier t_c peuvent s'intersecter au sein de celui-ci. Au-

[5] effectuant ainsi deux fois l'extraction de la surface de bord, ce qui n'est pas optimal mais garantit l'indépendance des différentes étapes de précalculs. Si les deux étapes sont réalisées de manière consécutives, alors l'extraction de la surface de bord peut n'être réalisée qu'une et une seule fois et les temps de calculs sont encore améliorés (cf. annexe B).

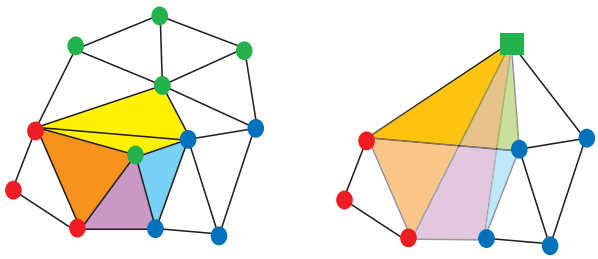


FIG. 2.22: **Exemple de condition non vérifiée pour la partition basée sur les coordonnées barycentriques**. La condition *C.iii* n'est pas vérifiée. En effet, une intersection géométrique a lieu entre les trois triangles fins orange, violet et bleu ciel avec les deux triangles jaunes (à gauche) lorsque le sommet grossier (carré) correspondant au cluster vert est conservé. L'extraction du maillage birésolution (à droite) impliquent que les deux triangles jaunes se retrouvent confondus alors qu'ils sont deux entités différentes et que le triangle obtenu est lui-même recouvert par les triangles orange, violet et bleu ciel. Cette violation est due à la non connexité des clusters et à la non unicité des représentants du triangle grossier.

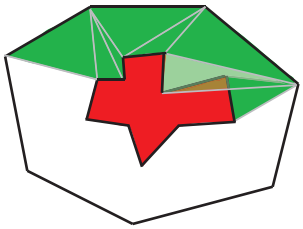


FIG. 2.23: **Intersection entre les tétraèdres de liens et les tétraèdres fins.** La condition *C.iii* n'est pas vérifiée. En effet, les tétraèdres de lien en vert intersectent en un endroit les tétraèdres fins en rouge. Le triangle causant l'intersection est plus claire afin de révéler celle-ci. Le surface du maillage grossier est représentée en noir. Cette intersection est causée par la non convexité du cluster $\overline{\tau}_c$ (en rouge).

quel cas, au moins un tétraèdre fin $t_f \in |t_c|$ (où $|t_c|$ est l'espace occupé par le tétraèdre t_c) correspondra à cette intersection et permettra de reconstruire t_c.

Néanmoins, l'existence d'un tel tétraèdre fin (ayant quatre sommets dont les images par $\wp$ sont deux à deux différentes) n'est pas garantie dans toutes les situations même si on a imposé que tout tétraèdre grossier contienne au moins un tétraèdre fin. En effet, garantir l'existence d'un tel tétraèdre consiste à imposer que trois frontières de clusters homéomorphes à des 2-variétés s'intersectent en au moins un tétraèdre fin. Ce problème trivial dans le cadre de maillage triangulaire puisqu'équivalent à l'intersection de deux 1-variétés en un point sur une 2-variété (ce qui est toujours réalisable), se révèle ainsi plus complexe dans le cadre des maillages tétraédriques.

En pratique, il est toujours possible de réaliser un post-traitement au calcul de $\wp$ en incorporant au cours de la construction de la partition une validation de l'injection des tétraèdres grossiers au sein du maillage fin. Si celle-ci n'est pas vérifiée, c'est-à-dire qu'il existe des tétraèdres t_c n'ayant pas de tétraèdre fin associé, un tétraèdre fin se trouvant au sein du tétraèdre grossier $t_c = (v_{c_i})$ (il en existe au moins un) peut être transformé en modifiant localement la définition de $\wp$ afin que $\overline{\wp}(t_f) = \{v_{c_i}\}$.

Le recours a un tel post-traitement s'est révélé en pratique, sur l'ensemble des tests que nous avons réalisés, un phénomène rare.

Limitations Bien que cette nouvelle partition permet de vérifier les conditions de validité topologique (condition (2.1)) et de validité géométrique faible (condition (2.2)), cette solution ne va pas permettre la vérification de la condition *C.iii* lors de l'extraction du maillage birésolution. En pratique, la vérification de cette troisième condition afin d'obtenir un maillage conforme dans toutes les situations n'est pas une obligation pour le bon déroulement de l'exploration de données. En effet, bien que la non-conformité peut conduire à des artefacts visuels lors de l'utilisation des techniques de visualisation usuelles, dans la pratique la fréquence de ceux-ci est minime au sein de l'ensemble des tests que nous avons effectués. La localité de ces possibles artefacts, au sein de la zone de jonction entre les deux résolutions, garantit que l'information essentielle au sein de la zone d'intérêt ne sera jamais détériorée et bien à pleine résolution. De plus, la rapidité de calculs d'une telle partition indépendante de tout processus de simplification reste un atout majeur.

Néanmoins, nous avons décelé certaines situations qui provoquent la violation de la condition *C.iii*. Leur fréquence d'apparition est fortement liée aux deux maillages d'entrée, pouvant ne jamais se produire ou, au contraire, concerner un certain nombre d'extractions dynamiques. Ces situations peuvent être facilement décelées au cours d'un post-traitement au calcul de $\wp$ en parcourant les tétraèdres fins. Ces situations sont les

suivantes :

- la connexité des clusters S_c illustrée au sein de la figure 2.21 (b) ;
- la non unicité du tétraèdre fin se projetant sur un tétraèdre grossier illustrée au sein de la figure 2.21 (c) ;
- la non convexité des clusters $\overline{\tau}_c$.

Ces situations peuvent ainsi causer des intersections entre différents tétraèdres de lien (figure 2.22) ou entre les tétraèdres fins et les tétraèdres de lien (figure 2.23).

Bien que la vérification de la troisième condition *C.iii* ne soit pas une nécessité pour le bon déroulement de l'exploration des données comme nous avons pu le remarquer en pratique, nous évoquons quelques premières esquisses de solutions au sein de la section suivante afin d'augmenter la fréquence d'une extraction d'un maillage birésolution conforme *via* la réalisation des trois conditions données en section 2.3.1.

3 Conclusion et Perspectives

Au sein de ce chapitre, nous avons détaillé l'élaboration d'une partition des sommets d'un maillage tétraédrique sous la contrainte d'une discrétisation de ce même espace à une résolution plus grossière.

Pour cela, nous avons, dans un premier temps, proposé un algorithme de simplification pour les grilles irrégulières tétraédriques permettant le traitement de plus de quarante millions de tétraèdres au sein d'un ordinateur possédant deux mégaoctets de mémoire vive. Cette simplification repose sur trois étapes : l'extraction de la surface de bord et sa simplification ; l'extraction des extrema locaux du champ scalaire ; et une tétraédralisation contrainte de Delaunay. Elle garantit ainsi la vérification d'un certain nombre de critères de qualité pour les tétraèdres grossiers générés.

Nous avons ensuite proposé la construction d'une partition des sommets fins sous la contrainte d'un maillage grossier en essayant de respecter trois conditions permettant de garantir la conformité du maillage birésolution qui sera extrait à partir de la définition de cette partition. Afin d'assurer ces conditions, nous avons défini la validité topologique et géométrique faible d'une partition.

La partition reposant sur l'utilisation des coordonnées barycentriques issues du plongement du maillage fin au sein du maillage grossier est la meilleure candidate car elle vérifie les validités topologique et géométrique faible. Le calcul a de plus une faible complexité temporelle (assurant ainsi des temps de précalculs faibles). Néanmoins, elle ne permet la réalisation d'un maillage conforme dans tous les cas. En pratique, l'absence de la conformité dans toutes les situations ne nuit aucunement à l'exploration des données. Cette partition est donc pleinement adaptée à notre problématique. Dans la suite de cette thèse, l'utilisation du mot *partition* fera référence à la partition étendue $\wp$ reposant sur les coordonnées barycentriques.

Une des forces de la solution proposée pour le calcul de la partition est sa généralité. En effet, suite à de premières investigations, une telle solution semble généralisable entre un maillage hybride fin (composé de tétraèdres, de pentaèdres et d'hexaèdres) et un maillage grossier tétraédrique. L'extraction du maillage birésolution serait donc réalisée en générant des cellules de lien hybrides en appliquant des règles de décomposition de ces cellules en fonction des différents cas possibles.

Nous avons aussi déjà exploré quelques pistes afin de vérifier la condition *C.iii* manquante permettant l'extraction d'un maillage conforme. D'après la figure 2.22, l'unicité du tétraèdre fin dont l'image par la surjection est un tétraèdre grossier comme la connexité des clusters semblent des prérequis pour limiter de telles intersections. Ces deux prérequis sont assurés par une partition reposant sur un algorithme de simplification par contraction d'arêtes.

Afin d'atteindre cet objectif tout en garantissant la validité topologique et géométrique faible, une solution envisageable serait de combiner l'approche barycentrique avec une approche par effondrements d'arêtes pour chaque ensemble de tétraèdres fins inclus dans l'espace occupé par un tétraèdre grossier. En choisissant un tétraèdre fin initial à quatre couleurs et en propageant cette information par effondrement d'arêtes pondérées par une erreur définie grâce aux coordonnées barycentriques, les garanties apportées par les deux solutions précédentes seraient ainsi assurées. L'inconvénient majeur de cette solution serait la nécessité de calculer des relations topologiques entre certains simplexes du maillage fin afin d'assurer la contraction d'arêtes. Cela augmenterait la consommation mémoire de l'algorithme.

Néanmoins, cela ne reste pas suffisant pour garantir la condition *C.iii* dans toutes les situations car elle ne garantirait pas la connexité des clusters. De futurs travaux sont encore nécessaires si l'on souhaite garantir la

conformité dans toutes les situations de notre maillage birésolution comme le garantit les approches usuelles multirésolution.

CHAPITRE 3

Bi-Résolution : Extraction Dynamique

Scientific visualization is concerned with exploring data and information in such a way as to gain understanding and insight into the data. This is a fundamental objective of much scientific investigation.

Ken W. BRODLIE - *Scientific Visualization : Techniques and Applications*, 1992

COMME vu précédemment au sein du chapitre 2, des algorithmes permettant de simplifier puis de construire des hiérarchies de niveaux de détails ont été proposés depuis une décennie afin d'extraire dynamiquement un maillage tétraédrique à précision variable. En effet, lors de la phase d'exploitation des résultats, phase qui consiste comme nous l'avons déjà précisé, à vérifier l'exactitude de la simulation ou à détecter les zones de possibles intérêts grâce à une exploration des données (cf. chapitre 1, section 3.1), la taille de celles-ci ne peut permettre cette exploration en temps-réel (cf. chapitre 1, section 3.2). Or l'interactivité de cette étape de visualisation est primordiale afin de faciliter le travail des chercheurs ou des ingénieurs en quête de la validation ou de la confrontation de leurs résultats [HS89].

Afin d'explorer interactivement les résultats, les schémas multirésolution affichent un maillage grossier tout en permettant d'extraire au sein de zones locales spécifiées par l'utilisateur, une fine granularité contenant l'ensemble des données initiales. La *conformité* du maillage assurée par les schémas multirésolution permet de plus d'assurer une continuité géométrique – et donc visuelle – entre les différentes granularités, clef essentielle pour comprendre un phénomène afin de mettre en évidence les causes et les effets d'un phénomène local au sein des tendances globales.

Dans ce chapitre, nous expliquons la mise en œuvre d'un schéma d'exploration de grosses grilles irrégulières tétraédriques utile pour l'exploitation de résultats issus de simulations numériques. Il repose sur une approche birésolution composée d'une résolution fine et d'une résolution grossière. Le maillage le plus fin sera utilisé au sein de zones d'intérêt définies par l'utilisateur alors que le maillage grossier sera conservé hors de celles-ci. La

construction d'un maillage unique liant les deux résolutions repose fortement sur les propriétés de la partition énoncées dans le chapitre 2 précédent.

Nous détaillons plus précisément la construction d'un tel maillage birésolution ainsi que la problématique de recherche de zones d'intérêt dans laquelle elle s'insère au sein de la section 1. Puis, nous expliquons les structures de données ainsi que les algorithmes mis en place afin d'assurer une exploitation interactive à la fois au sein du processeur principal de l'ordinateur (section 2) mais aussi des nouvelles générations de cartes graphiques (section 3). Nous proposons enfin une approche en mémoire externe (*out-of-core*) dans la continuité des solutions proposées au sein du chapitre 2 précédent en section 4. Nous concluons et discutons de futurs travaux dans la section 5.

1 Schéma Birésolution : Problématique et Construction

Au sein de cette section, nous développons de manière théorique l'implantation de l'extraction d'un maillage birésolution. Nous revenons tout d'abord plus précisément sur les raisons de l'élaboration d'un tel schéma en section 1.1 en consacrant un rapide état de l'art sur les méthodes interactives permettant l'extraction de zones d'intérêt. Puis, nous exposons la construction du maillage birésolution en section 1.2.

1.1 Extraction de zones d'intérêt

L'extraction de connaissances utiles à la compréhension d'un phénomène simulé est une nécessité lors de l'exploitation des données. Néanmoins, cette opération est complexe même pour des utilisateurs expérimentés. En effet, la compréhension visuelle d'un ensemble de données repose sur une connaissance préalable de ce qui est visualisé comme par exemple la nature du champ simulé : température, pression, vitesse de déplacement, cisaillement,... L'ingénieur ou le chercheur projette donc ses connaissances – parfois maigres – pour guider son exploration en extrapolant sa propre vision du résultat au sein de la simulation. Naturellement, le choix de la technique de visualisation : extraction d'isosurfaces, rendu volumique, *streamlines* ; aura un impact sur cette compréhension. Elle est d'autant facilitée si l'utilisateur est capable rapidement de détecter des régions particulières, dite *d'intérêt* : point chaud, vortex, basse pression, haute probabilité,... L'utilisateur final a donc besoin de pouvoir explorer **interactivement** les données afin de faciliter la recherche de telles zones d'intérêt souvent localisées (cf. chapitre 1, section 5.3) dans l'optique d'accélérer sa propre compréhension du phénomène [HS89].

Les schémas multirésolution construits sur les maillages tétraédriques tentent de répondre à cette problématique clef d'exploration interactive. En définissant une région locale géométrique à haute résolution, les utilisateurs peuvent ainsi explorer interactivement la simulation afin d'en percer les problèmes ou de la valider. L'interactivité est d'autant plus importante qu'elle permet des allers et retours facilités entre des zones d'intérêt. Ces mouvements rendent possible la réalisation d'une reconstruction mentale plus complète de la simulation. Ces schémas aident donc à comprendre plus rapidement les connectiques entre les différentes zones d'intérêt et ainsi à reconstruire le phénomène dans sa globalité [Bou09].

Ces approches multirésolution pour les grilles irrégulières peuvent ainsi s'inscrire dans l'ensemble des techniques dites *Focus+Contexte* qui ont été développées au cours de la dernière décennie principalement pour les grilles régulières. Ces techniques interactives se concentrent sur l'extraction de zones d'intérêt en essayant, pour les plus récentes, d'introduire des procédés automatiques de détection de ces régions. Elles s'aident pour cela d'un procédé de segmentation réalisé en prétraitement permettant, pour les données médicales, de séparer les différents matériaux. Nous les détaillons plus précisément ci-après.

1.1.1 Techniques Focus+Contexte

Les techniques *Focus+Contexte* (F+C) cherchent à mettre en emphase des zones définies par l'utilisateur. Dans le cadre de cette thèse, nous nous restreignons aux méthodes imitant les effets optiques que produiraient une loupe sur une feuille de papier ou les effets matériels que créeraient une extrusion de matière. Elles reposent ainsi dans leur majorité sur la définition d'une région locale dans l'espace spatiale : sphère, cube,... ; ou dans l'espace image : cercle, carré,... ; afin d'agir respectivement sur la géométrie ou les pixels. Au sein et autour de

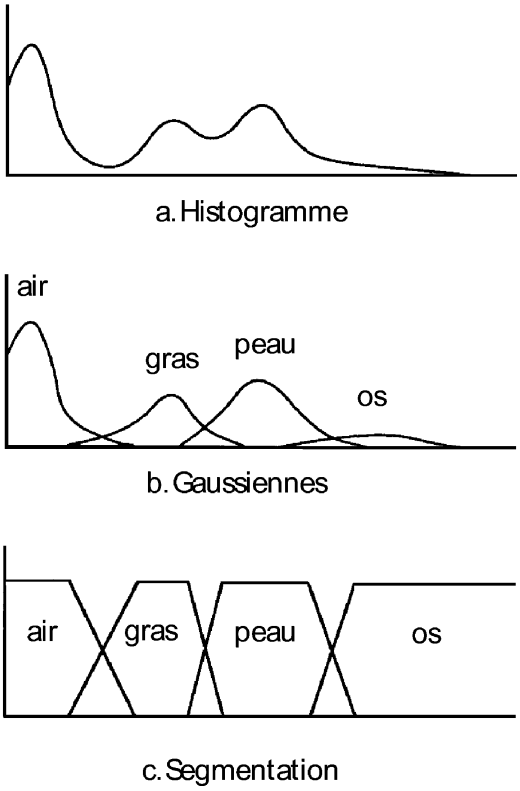


FIG. 3.1: **Étapes de segmentation de données médicales régulières** issu de [DCH88]. (a) L'histogramme du champ d'attributs est calculé. (b) Des gaussiennes sont associées à l'histogramme en fonction des connaissances anatomiques. (c) La segmentation est réalisée sur le champ scalaire.

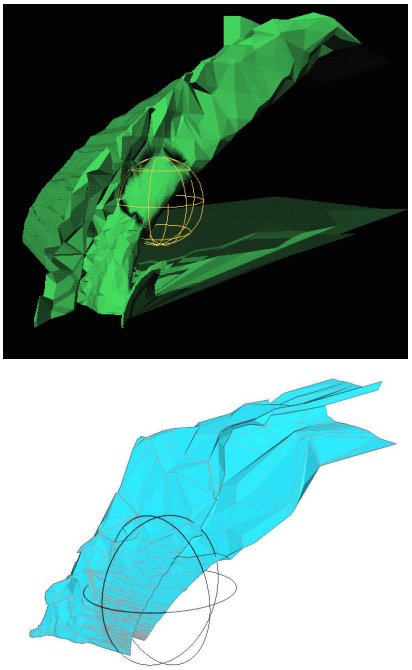


FIG. 3.2: **Comparaison entre *MagicSphere* [CDFM+94] et *BiRes* (notre méthode).** On extrait dans les deux cas une isosurface locale fine au sein de la sphère d'intérêt et une isosurface grossière à l'extérieur pour l'ensemble de données *Blunt Fin*. En haut, *MagicSphere*. En bas, notre méthode *BiRes*. On remarque l'usage de la transparence pour créer un raccord diffus entre les deux résolutions n'empêchant nullement l'apparition de trous au sein de l'isosurface dans l'image du haut. Notre méthode assure l'extraction d'une unique isosurface à deux résolutions.

cette loupe, les rendus ainsi que la précision des détails sont souvent spécifiques aux besoins de l'utilisateur et peuvent mêler traitement optique de grossissement ou extrusion, rendu usuel : isosurfaces, rendu volumique, *streamlines* ; et rendu non photoréaliste inspiré des dessins industriels ou anatomiques. Pour une taxinomie plus détaillée, le lecteur peut se référer à [Hau05, CB04].

Ces techniques ont été principalement développées pour les données régulières médicales et de simulation de flux. Le principe fondateur des méthodes F+C a été introduit via l'outil *MagicLens* (lentille magique) [BSP+93] proposant des formes et des filtres de post-traitement en espace image pour définir des lentilles permettant l'exploration interactive de dessins.

Cette idée fut ensuite généralisée à la visualisation d'isosurfaces avec la *MagicSphere* de Cignoni [CDFM+94] qui permet d'extraire une résolution fine au sein d'une zone locale géométrique sphérique et une résolution grossière autour. Les deux maillages s'intersectent au niveau de la surface de la sphère. Cet algorithme n'offre donc pas une reconstruction entre les deux résolutions surfaciques pour des raisons d'interactivité. À la place, un mélange par transparence est assuré afin de simuler l'unicité et réduire les artefacts visuels dus aux intersections entre les deux résolutions. La figure 3.2 illustre le résultat obtenu *via* cette technique et offre une comparaison avec notre approche reconstruisant à la volée un unique maillage sur le même ensemble de données.

Ce concept fut ensuite appliqué aux textures tridimensionnelles [LHJ01] grâce à une distorsion locale des coordonnées de textures résultant à des grossissements locaux. Le focus spatial est aussi largement utilisé en visualisation de champs d'attributs vectoriels, notamment en modifiant localement le nombre de lignes

de courant affichées [FG98, MTHG03]. Une continuité entre le focus et le contexte a aussi été récemment proposée [BKKW08] pour les champs de particules grâce à des *morphings* de transparences et de formes.

Reposant sur des bases d'optique géométrique, la métaphore de loupe a été implantée sur la carte graphique grâce à un lancer de rayons qui simule leur déviation comme le ferait une loupe réelle [WZMK05]. Cette solution permet ainsi de réaliser un zoom optique au sein de la loupe tout en conservant l'information sans grossissement à l'extérieur. Une zone de transition est assurée afin de garantir la continuité visuelle. Néanmoins cette zone est compressée puisque l'espace contenu dans la loupe subit un grossissement dans l'espace image provoquant l'occupation d'un nombre plus important de pixels par cette région d'intérêt.

Les méthodes multirésolution comme les approches précédentes s'appuient principalement sur la définition géométrique d'un focus dans l'espace objet ou image, le focus étant assimilé principalement à une sphère ou un cercle. Néanmoins, il est aussi possible de définir le focus sur le champ des attributs. Pour cela, il est possible d'effectuer un certain nombre de précalculs reposant sur des connaissances préalables. Par exemple, les acquisitions médicales peuvent être découpées spatialement en différentes zones d'intérêt représentant les différents organes en utilisant les connaissances anatomiques des médecins. On parle alors de *segmentation* des données. Les étapes de segmentation d'un champ régulier sont illustrées au sein de la figure 3.1. Dans un premier temps, un histogramme des données est calculé. Cet histogramme est approché *via* un certain nombre de gaussiennes, une par matériaux à identifier dans l'idéal. Ces gaussiennes, confrontées aux statistiques anatomiques, permettent enfin grâce à une correspondance intelligente l'association d'un matériau à chaque pic et donc à un intervalle scalaire. Ainsi, [WZMK05] l'utilise afin de définir des loupes englobant des intervalles scalaires.

La segmentation n'est pas l'unique procédé possible. [VFSG06] réalise une modulation de la transparence pour le rendu volumique en fonction d'une mesure d'information mutuelle, ce qui empêche des occlusions du focus par le contexte. Cette modulation peut être couplée avec une mesure de saillance [KV06]. Pour des nuages de points, [DGH03] et [Gas04] permettent à l'utilisateur de choisir ces zones d'importance sur des histogrammes représentant les répartitions des différents attributs. Des intersections ou des unions entre diverses valeurs d'attributs sont ainsi représentées en utilisant une colorisation des points appartenant au focus alors que le contexte reste grisé.

1.1.2 Discussion

L'ensemble des techniques Focus+Contexte comme les approches multirésolution tentent de répondre aux mêmes problématiques : l'extraction de connaissances au sein d'ensembles de données complexes. Leur but est donc de faciliter la compréhension de l'utilisateur pour qu'il puisse exploiter ses résultats. Néanmoins, le support géométrique est des plus différents. Les premières reposent principalement sur des grilles régulières (données médicales, simulation de flux) voire des nuages de points alors que les secondes s'appuient sur des maillages irréguliers.

Ainsi, les techniques reposant principalement sur des données médicales régulières sont difficilement transposables aux maillages irréguliers. L'absence de segmentation empêche l'utilisation des méthodes reposant sur une telle extraction du savoir. Bien que l'extraction d'intervalles d'attributs soit possible, celle-ci reste souvent insuffisante dans le cadre de simulations complexes où l'utilisateur, au premier abord, est incapable de spécifier de tels intervalles. L'algorithme proposé par [WZMK05] nécessite l'implantation d'un lancer de rayons pour les grilles irrégulières sur la carte graphique afin d'assurer l'interactivité de l'exploration. De telles solutions existent – comme nous l'évoquerons au sein du chapitre 4 ; mais, de par leur structure, sont difficilement transposables directement. Les chercheurs ou ingénieurs lors de la phase d'exploitation peuvent de plus désapprouver une distorsion de l'image qui pourrait modifier fortement des caractéristiques clefs de la structure visualisée comme sa surface de bord.

Pour les maillages irréguliers, la solution proposée par [CDFM+94] propose une extraction au sein de l'espace géométrique d'une zone locale pour la visualisation d'isosurfaces. Néanmoins, l'absence d'un maillage unique empêche la généralisation de cette méthode à d'autres méthodes de rendu comme un rendu volumique direct ou des lignes de courant. En effet, l'apparition d'un trou géométrique entre les deux résolutions peut entraîner des artefacts comme des ruptures visuelles qui pourraient se révéler dérangeantes pour l'utilisateur. La solution choisie d'une modulation de transparence n'est de plus pas satisfaisante dans le cadre de la visualisation d'isosurfaces. En effet, ce floutage simple ajoute du bruit sous la forme d'un nuage vaporeux qui va

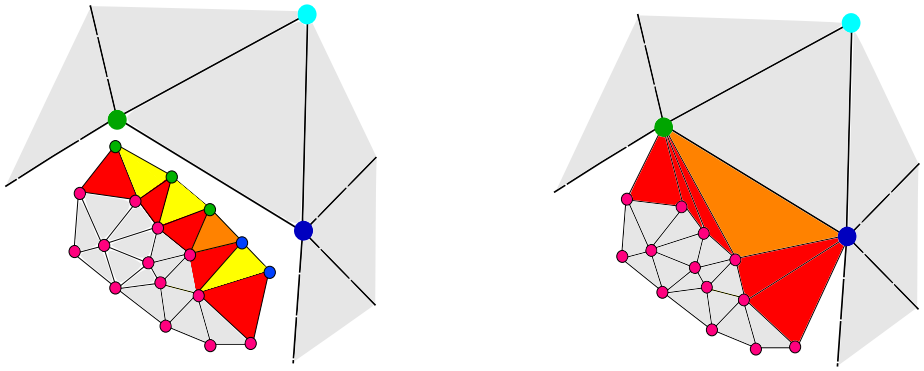


FIG. 3.3: **Transformation des cellules de lien pour un maillage triangulaire (pour des raisons de clarté)**. À gauche : le cluster rose appartient à la zone d'intérêt : il est *actif*. L'ensemble des triangles fins actifs n'est pas modifié. Les clusters vert, bleu et cyan sont inactifs. Les triangles *de lien* subissent une transformation. On distingue trois types de triangles de lien : ceux ayant deux sommets actifs et un inactif (en rouge) ; ceux ayant un sommet actif et deux sommets inactifs appartenant au même cluster (en jaune) qui deviendront des simplexes dégénérés ; et ceux ayant un sommet actif et deux sommets inactifs appartenant à deux clusters différents (en orange). À droite : le résultat obtenu en utilisant la surjection $\wp$. Les triangles de lien rouges et oranges sont étirés. Les triangles jaunes sont rejetés. Pour les maillages tétraédriques, les tétraèdres de lien subissent les mêmes transformations. Les cellules devenant des d-faces ($d < 3$) sont rejetées.

dissimuler le comportement de l'isosurface dans la zone de jonction. De brusques variations au sein de cette zone, se traduisant par exemple par des courbures fortes de la surface, pourraient ainsi être indécelables.

Les méthodes Focus+Contexte sont donc peu adaptées aux problématiques liées à l'exploitation des résultats au sein de grilles irrégulières. Au contraire, les approches multirésolution le sont. La définition dans l'espace objet d'une zone d'intérêt manipulable par l'utilisateur permet ainsi une exploration libre au sein du maillage sans aucune restriction sur les techniques de visualisation utilisables. Néanmoins, contrairement à leurs consœurs dites Focus+Contexte, elles n'ont jamais été transposées sur la carte graphique ce qui limite leur interactivité.

Dans cette thèse, nous proposons donc un nouveau schéma multirésolution reposant sur deux granularités d'une même simulation, correspondant aux niveaux de détails extrêmes des schémas de multirésolution usuels. Pour cela, nous n'utilisons que la partition décrite dans le chapitre 2 précédent. Ce schéma va permettre des gains temporels et mémoriels non négligeables par rapport aux précédentes approches multirésolution et son implantation sur carte graphique. Dans la suite de cette section, nous détaillons plus précisément la construction du maillage birésolution à partir de cette partition.

1.2 Extraction d'un maillage birésolution

Notre approche, nommée *BiRes*, repose sur l'extraction dynamique d'un maillage birésolution. Elle nécessite en entrée un maillage tétraédrique à haute résolution et une partition des sommets fins contrainte par un maillage tétraédrique à basse résolution. L'extraction d'une telle partition à partir de maillages à deux résolutions différentes a fait l'objet du chapitre 2. Afin d'extraire localement la zone à haute résolution, une erreur d'approximation dynamique est tout d'abord définie en section 1.2.1. L'union entre les deux résolutions est expliquée en section 1.2.2 et quelques propriétés vérifiées par le maillage birésolution sont explicitées en section 1.2.3. Nous discutons enfin cette approche (section 1.2.4).

1.2.1 Erreur d'approximation dynamique

L'extraction dynamique du maillage birésolution est réalisée *via* la définition d'une erreur dynamique. Cette erreur repose sur une distance métrique d. Par exemple, on peut la définir de la manière suivante :

Définition 3.1 Boule d'intérêt : On définit la zone locale d'intérêt comme une boule $\mathcal{B}_l$ telle que :

$$\mathcal{B}_l = \left\{x \in \mathbb{R}^3 | d(x,o) < R\right\}, o \in \mathbb{R}^3$$

L'utilisateur a ainsi le contrôle sur trois paramètres distincts : la position centrale o de la zone d'intérêt, sa taille R et sa forme d. Dans la pratique, nous utilisons pour d la distance euclidienne d_e pour obtenir une sphère mais rien n'empêche par exemple d'utiliser la distance infinie $d = d_\infty$ pour définir un cube ou toute autre métrique.

Au cours de son exploration des données, l'utilisateur peut ainsi déplacer le centre d'intérêt o et faire varier son rayon d'extraction R afin d'exploiter les résultats. Par abus de langage, on notera la boule d'intérêt $\mathcal{B}$ dans la suite de ce manuscrit.

1.2.2 Extraction des tétraèdres de lien

La définition de la boule d'intérêt $\mathcal{B}$ (cf. définition 3.1) va permettre de distinguer deux types de clusters de sommets S_v issus de la partition $\wp$: ceux ayant une intersection non vide avec la zone d'intérêt ($S_v \cap \mathcal{B} \neq \emptyset$) et leurs complémentaires ($S_v \cap \mathcal{B} = \emptyset$). Nous appelons respectivement les premiers *actifs* que l'on notera S_v^{actif} et les seconds *inactifs* que l'on notera $S_v^{inactif}$. De plus, nous nommons aussi *actif* (respectivement *inactif*) l'ensemble des sommets fins v_f appartenant à un cluster actif (respectivement inactif).

Suivant cette dénomination des clusters, et en utilisant $\bar{\wp}$, l'extension de $\wp$ aux tétraèdres (cf. chapitre 2, section 2.4), on peut aussi caractériser les cellules. Ainsi, les tétraèdres peuvent être de trois types :

- *actifs* : tétraèdres possédant quatre sommets actifs ;
- *de lien* : tétraèdres ayant au moins un sommet actif et au moins un sommet inactif ;
- *inactifs* : tétraèdres possédant quatre sommets inactifs.

Pour construire le maillage birésolution, nous définissons :

- $\mathcal{F}$: ensemble des tétraèdres fins actifs. On peut définir aussi $\mathcal{F}$ comme l'union des τ_c^{actif} avec $t_f \in \tau_c^{actif}$ si et seulement si $t_f \in \tau_c$ (cf. equation (2.1)) et t_f est actif ;
- $\mathcal{G}$: nerf des ensembles des tétraèdres fins inactifs $\tau_c^{inactif}$. On définit ces ensembles tels que : $t_f \in \tau_c^{inactif}$ si et seulement si $t_f \in \tau_c$ (cf. équation (2.1)) et t_f est inactif ;
- $\mathcal{L}$: ensemble des tétraèdres de lien.

Par définition, on a l'ensemble des tétraèdres fins T_f vérifiant $T_f = \mathcal{L} \bigcup (\cup \tau_c^{actif}) \bigcup (\cup \tau_c^{inactif})$.

Afin d'obtenir un maillage unique réunissant les ensembles $\mathcal{F}$ à haute résolution et $\mathcal{G}$ à basse résolution, les tétraèdres fins de lien $\mathcal{L}$ sont transformés en utilisant la surjection $\wp$. Les sommets fins actifs restent inchangés alors que les sommets fins inactifs sont projetés sur le sommet grossier qui est leur image par $\wp$. Une telle transformation peut provoquer l'effondrement de plusieurs sommets inactifs sur le même sommet grossier. De tels tétraèdres sont *dégénérés*. Ils sont enlevés lors du processus de reconstruction du lien. La figure 3.3 illustre les transformations subies par les cellules de lien dans le cadre de maillages triangulaires.

La construction du maillage birésolution Σ_b est définie par :

$$\Sigma_b = \mathcal{F} \cup \mathcal{G} \cup \bar{\wp}(\mathcal{L}) \tag{3.1}$$

Il est extrait selon l'algorithme suivant :

***ALGORITHME* 2**

```
Pour tout tétraèdre fin t_f = (v_{f_i})_{i∈[1..4]} ∈ T_f
  Calculer le nouveau tétraèdre t'_f = (w_{f_i}) tel que
  Pour tout sommet w_{f_i},
     Si v_{f_i} est actif,
     Alors garder w_{f_i} = v_{f_i} ∈ V_f
     Sinon w_{f_i} = ℘(v_{f_i}) ∈ V_c
  Si t'_f est dégénéré, le rejeter
```

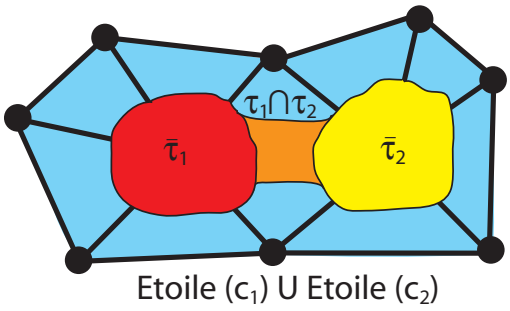


FIG. 3.4: **Intersection vide entre maillage fin et grossier.** Deux clusters sont actifs : S_1 et S_2 associés respectivement aux sommets grossiers c_1 et c_2. L'ensemble $\bar{\tau}_1$ est représenté en rouge, $\bar{\tau}_2$ en jaune, l'intersection non vide $\tau_1 \cap \tau_2$ en orange. L'union des étoilés des sommets grossiers actifs est représenté en bleu ciel. On a $\tau_1^{actif} \cup \tau_2^{actif} = \bar{\tau}_1 \cup \bar{\tau}_2 \cup (\tau_1 \cap \tau_2)$ par définition des τ_c^{actif}. La propriété suivante est ainsi vérifiée : $|\tau_1^{actif} \cup \tau_2^{actif}| \subset |Etoile\ (c_1) \cup Etoile\ (c_2)|$.

1.2.3 Propriétés du maillage birésolution

La construction du maillage birésolution Σ_b est réalisée d'après un maillage fin Σ_f et une partition des sommets fins contrainte par un maillage grossier Σ_c déduite de la surjection $\wp$. Le maillage birésolution vérifie un certain nombre de propriétés reposant sur celles de $\wp$ mise en évidence au sein du chapitre 2 précédent. Plus particulièrement, on rappelle que la surjection $\wp$ définie selon les coordonnées barycentriques (chapitre 2, section 2.6), vérifie la validité topologique (équation (2.2)) et la validité géométrique faible (équation (2.3)).

Nous énonçons ces propriétés ci-après.

Propriété 3.1 *$\mathcal{G}$ est un sous-ensemble du maillage grossier Σ_c.*

On suppose qu'il y a k clusters inactifs et l clusters actifs tels que $k + l = card\ V_c$. $\mathcal{G}$ est défini comme le nerf des ensembles $\tau_c^{inactif}$ de tétraèdres fin inactifs. On a donc :

$$\begin{array}{ccc} \bigcup_{i=1}^{k} \tau_{c_i}^{inactif} & \subseteq & \bigcup_{j=1}^{card\ V_c} \tau_j \\ Nerf\ (\tau_{c_1}^{inactif}, \ldots, \tau_{c_k}^{inactif}) & \subseteq & Nerf\ (\tau_1, \ldots, \tau_{card\ Vc}) \end{array}$$

Comme $\wp$ garantit que $Nerf\ (\tau_1, \ldots, \tau_{card\ Vc}) = \Sigma_c$ car $\wp$ est valide topologiquement (cf. equation (2.2)), on obtient :

$$\left(Nerf\ (\tau_{c_1}^{inactif}, \ldots, \tau_{c_k}^{inactif}) = \mathcal{G}\right) \subseteq (Nerf\ (\tau_1, \ldots, \tau_{card\ Vc}) = \Sigma_c)$$

donc $\mathcal{G} \subseteq \Sigma_c$. ■

Propriété 3.2 *Il n'y a pas d'intersection entre le maillage grossier $\mathcal{G}$ et le maillage fin $\mathcal{F}$ composant le maillage birésolution Σ_b (condition C.ii définie au chapitre 2, section 2.3.1) : $|\mathcal{G}| \cap |\mathcal{F}| = \emptyset$ où $|\Sigma|$ est l'espace occupé par le complexe simplicial Σ.*

$\wp$ est valide géométriquement faible. Donc selon l'équation (2.3) :

$$|\bar{\tau}_c| \subset |Etoile\ (c)|$$

où c est un sommet grossier et $\bar{\tau}_c = \bar{\wp}^{-1}(c)$ l'ensemble de tous les tétraèdres fins dont les sommets ont pour image par $\wp$ le sommet c.

Tout d'abord, par définition, on a $\mathcal{F} = \bigcup \tau_c^{actif}$. On a alors l'inclusion suivante illustrée par la figure 3.4, assurée par la validité géométrique faible de $\wp$ (équation (2.3)) :

$$|\mathcal{F}| \subset |\bigcup_{S_c^{actif}} Etoile\ (c)|$$

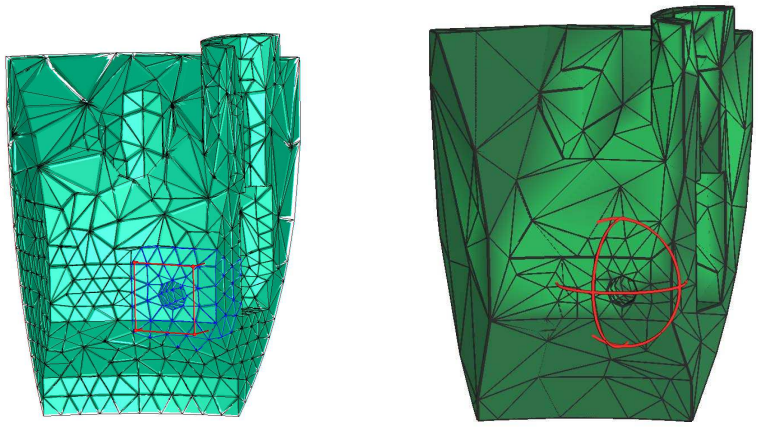


FIG. 3.5: **Localité de l'extraction.** À gauche en utilisant *MT* [CDFM+04]. À droite avec notre méthode *BiRes*. Contrairement aux précédentes approches multirésolution, notre raffinement est guidé par le maillage grossier et non par les dépendances entre les opérations locales de décimation. Il en résulte que la région fine extraite est plus localisée par rapport à la définition de la boule d'intérêt.

c'est-à-dire que $\mathcal{F}$ est inclus dans l'espace occupé par l'union des étoilés des sommets grossiers c dont les clusters S_c par l'image inverse de $\wp$ sont actifs.

De plus, toujours par définition, on a :

$$\mathcal{G} = Nerf\ (\tau_{c_1}^{inactif}, \ldots, \tau_{c_k}^{inactif})$$

En utilisant $\wp$ on peut définir l'ensemble T_c^{actif} des tétraèdres grossiers de Σ_c vérifiant :

$$T_c^{actif} = \{t_c = (v_{c_i}) \in T_c / \exists i,\ S_{c_i} \text{ est actif } \}$$

pour en déduire une autre définition pour $\mathcal{G}$:

$$\mathcal{G} = \Sigma_c \setminus T_c^{actif}$$

Cela veut dire que $\mathcal{G}$ correspond au maillage grossier initial Σ_c privé de tous les tétraèdres grossiers dont au moins un des sommets a pour image inverse de $\wp$ un cluster actif. On a donc $\mathcal{G} \cap T_c^{actif} = \emptyset$.

Or par définition des étoilés d'un sommet (cf. définition 1.3), on peut en déduire les inclusions suivantes :

$$|\bigcup_{S_c^{actif}} Etoile\ c| \subseteq |T_c^{actif}| \text{ donc } |\mathcal{F}| \subset |T_c^{actif}|$$

Cela signifie donc que : $|\mathcal{G}| \cap |\mathcal{F}| = \emptyset$. Il n'y a pas d'intersections géométriques entre les deux résolutions. ■

On rappelle que Σ_b n'est pas un complexe simplicial dans tous les cas comme nous l'avons expliqué lors de la définition de la surjection $\wp$ au sein du chapitre 2, section 2.6. En effet, elle ne garantit pas l'absence d'intersections entre les tétraèdres de lien et les tétraèdres fins et grossiers pour toute extraction dynamique.

1.2.4 Discussion

Notre proposition d'extraire un maillage birésolution a plusieurs avantages non négligeables.

Premièrement, la construction d'un tel maillage ne repose que sur la connaissance d'une partition des sommets fins. Elle n'impose donc nullement la construction d'une structure de dépendances (DAG ou forêt d'arbres binaires, cf. chapitre 2, section 1.2.2) consommatrice en mémoire. Elle permet ainsi de remplacer une structure de données nécessitant en moyenne $40n$ en place mémoire par une surjection nécessitant seulement n où $n = card\ V_f$ est le nombre de sommets fins.

Deuxièmement, la mise-à-jour n'est pas implantée grâce à l'utilisation d'un front actif nécessitant l'utilisation de deux piles à priorités afin de garantir le bon ordonnancement des opérations locales. Elle est donc

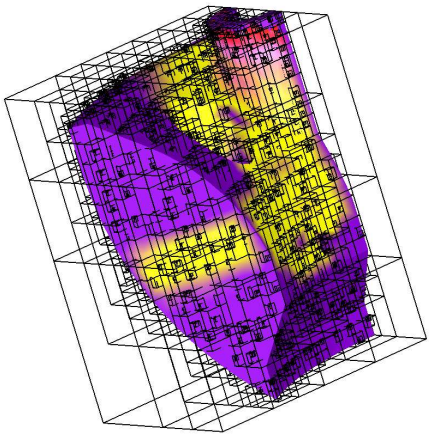


FIG. 3.6: **Octree construit sur un maillage tétraédrique**. La granularité de l'octree est adaptée localement en fonction de la présence ou non de tétraèdres fins.

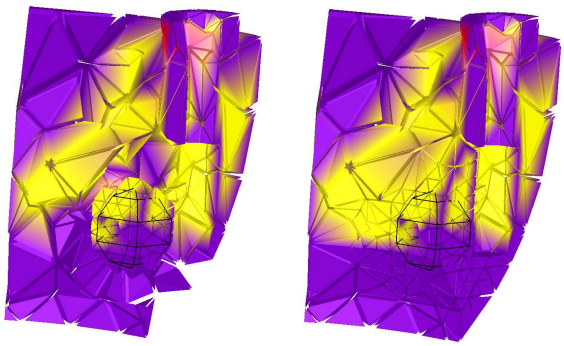


FIG. 3.7: **Extraction d'un maillage birésolution tétraédrique sur l'ensemble *SPX*.** Le champ scalaire est issu d'une simulation de l'écoulement d'un liquide de refroidissement au sein du réacteur Super Phénix et représente la vitesse de ce flux. La coloration des tétraèdres est basée sur le champ scalaire. À gauche, le maillage grossier $\mathcal{G}$ et le maillage fin $\mathcal{F}$ composé uniquement des tétraèdres se trouvant au sein de la boule d'intérêt. À droite, le maillage birésolution.

plus simple à réimplanter et à insérer au sein de logiciels de visualisation existants. Comme nous le détaillerons dans la suite de ce chapitre, cette solution garantit aussi des taux d'extraction jamais atteints par les précédentes approches.

Enfin, le découpage en clusters implique une localité spatiale de l'extraction fine restreinte aux frontières des clusters actifs. La propagation spatiale du raffinement non contrôlée à cause des dépendances entre les opérations locales de décimation sur lesquelles repose la majorité des précédentes approches (cf. chapitre 2, section 1.2.3) est ainsi évitée au sein de notre approche, autre garantie de son efficacité. Le raffinement fin est ainsi guidé par la définition du maillage grossier grâce à la vérification de la condition de validité géométrique faible par la surjection définissant la partition (chapitre 2, section 2.6). La figure 3.5 illustre la comparaison entre notre méthode et *MT* [CDFM+04].

2 Accélération de l'extraction sur le processeur central

L'extraction du maillage birésolution pourrait être implantée directement selon l'algorithme 2 fourni précédemment. Mais un tel choix impose le parcours de la résolution la plus fine à chaque mise à jour de la zone d'intérêt. Comme notre approche se doit d'être plus efficiente que l'utilisation d'un maillage monorésolution, cette solution ne remplit pas cet objectif. En effet, la complexité d'une telle implantation de l'extraction serait équivalente à traiter directement le maillage fin – même si, suite à l'extraction, le nombre de primitives envoyées au pipeline graphique est fortement inférieur au nombre de cellules contenues au sein de la résolution maximale.

Nous proposons donc des améliorations de l'algorithme afin de garantir la complexité : $O(card\ \mathcal{F} + card\ \mathcal{G})$ pour l'extraction de notre maillage birésolution où $\mathcal{F}$ et $\mathcal{G}$ sont respectivement les tétraèdres fins et grossiers constituant le maillage birésolution. Pour assurer le bon déroulement de cette optimisation, le calcul des relations d'adjacence pour chaque cellule est nécessaire en plus d'une structure indexée. Ce calcul est réalisé en prétraitement et permet d'obtenir pour chaque tétraèdre les indices de ses quatre tétraèdres voisins s'ils existent.

Afin de garantir cette faible complexité temporelle, nous avons mis en place des structures de données afin d'extraire rapidement la zone d'intérêt fine contenue au sein de la boule d'intérêt (section 2.1) mais aussi réaliser efficacement sa mise-à-jour en les couplant avec la cohérence temporelle (section 2.2). Nous détaillons ensuite l'extraction complète du maillage birésolution au sein du processeur central (section 2.3) avant d'en calculer la complexité et de discuter nos résultats (section 2.4). Un exemple du résultat obtenu par notre algorithme est donné au sein de la figure 3.7.

2.1 Extraction de la zone d'intérêt

L'erreur métrique dynamique définit une boule d'intérêt $\mathcal{B}(o,R)$ de centre $o \in \mathbb{R}^3$ et de rayon R – cf. Définition 3.1 – au sein de laquelle la résolution fine est extraite. Afin d'extraire le maillage fin inclus dans la boule d'intérêt $\mathcal{B}$, plusieurs étapes sont nécessaires : la localisation du centre d'intérêt o dans le maillage fin puis l'extraction des tétraèdres fins inclus dans $\mathcal{B}$. Cette étape permet ainsi de déduire la liste des clusters actifs qui permettront par la suite d'extraire l'ensemble des tétraèdres actifs.

Localiser le centre d'intérêt Notre première problématique est de localiser avec le plus faible coût possible (c'est-à-dire négligeable dans le procédé) le centre d'intérêt o de la boule afin d'extraire par la suite l'ensemble des clusters actifs de tétraèdres.

Pour cela, afin de localiser o rapidement sans parcourir tout le maillage fin, un *octree* est utilisé. Cette structure de données est construite sur les tétraèdres fins et stocke dans ses feuilles le barycentre des tétraèdres qu'elles contiennent. La profondeur de l'octree est choisie de manière à ce que les feuilles ne contiennent qu'un seul tétraèdre. Néanmoins, si la taille mémoire est limitée (ou inversement si le maillage est trop massif), la profondeur peut être restreinte de façon à ce que les feuilles contiennent quelques dizaines de tétraèdres sans perte de performances. La figure 3.6 illustre la construction de l'octree sur un maillage tétraédrique.

Pour trouver le tétraèdre fin le plus proche du centre o de la boule d'intérêt $\mathcal{B}$, l'octree est traversé au début de la phase d'extraction. Le tétraèdre extrait est nommé *source*. De plus, ses clusters de sommets, images selon $\overline{\wp}$, sont marqués comme actifs.

Extraction des tétraèdres inclus dans la boule d'intérêt Suite à l'extraction du tétraèdre source, les clusters actifs (c'est-à-dire ayant une intersection non vide avec $\mathcal{B}$, cf. section 1.2.2) doivent être extraits avec un coût en pire cas minimal.

Pour cela, les relations d'adjacence entre les cellules sont utilisées afin d'étendre la région à partir de la source. Un parcours en profondeur permet de réaliser cette extension : on parle alors d'un algorithme par croissance de régions (ou *region-growing*). Durant ce parcours, chaque nouveau tétraèdre fin atteint est testé afin de vérifier si au moins un de ses sommets appartient à la boule d'intérêt. La liste des clusters actifs est mise à jour en conséquence si nécessaire. Tout tétraèdre détecté comme actif est marqué afin d'éviter des ajouts intempestifs. L'algorithme de croissance de région est stoppé lorsque tous les tétraèdres fins appartenant à la boule d'intérêt ont été inclus.

Lors de cette extraction des tétraèdres fins, nous distinguons deux types de cellules illustrés au sein de la figure 3.8 : les tétraèdres actifs *de bord* intersectant la sphère d'intérêt et les tétraèdres actifs *internes* strictement inclus dans la boule d'intérêt $\mathcal{B}$. Les tétraèdres actifs internes sont stockés au sein d'une table de hachage dont la clef est basée sur le modulo de leur indice – ce qui permet de contrôler la taille de cette table à la fois en fonction du nombre de tétraèdres fins et en fonction de la place mémoire. Les tétraèdres actifs de bord sont, quant à eux, insérés dans une liste dite de bord.

L'algorithme par croissance de régions peut ne pas extraire l'ensemble des tétraèdres fins intersectant $\mathcal{B}$ lorsque celle-ci intersecte plusieurs composantes non connexes. Pour résoudre ce problème, des parcours supplémentaires de l'octree sont réalisés afin de détecter l'existence de tétraèdres à l'intérieur de la boule d'intérêt qui n'auraient pas été marqués actifs. Des tétraèdres sources sont ainsi ajoutés, un par composante connexe existante, et le même procédé de croissance par régions est appliqué à partir de chaque nouveau tétraèdre source.

2.2 Mise-à-jour de la zone d'intérêt

Au cours de l'exploitation des résultats, la boule d'intérêt va être déplacée au sein du maillage. L'extraction de notre maillage birésolution Σ_b doit donc prendre en considération ce déplacement afin d'assurer une faible complexité temporelle. Supposons que la zone d'intérêt $\mathcal{B}_t$ est déplacée par l'utilisateur au cours de son exploration au sein du maillage tétraédrique vers $\mathcal{B}_{t+1}$. Ce mouvement, dans la plupart des cas, va décaler $\mathcal{B}_{t+1}$ dans un espace proche de $\mathcal{B}_t$. On peut donc supposer que les différences entre deux maillages extraits Σ_b^t et Σ_b^{t+1} seront mineures. L'extraction réalisée pour Σ_b^t sera donc proche de celle qu'il faudrait réaliser pour Σ_b^{t+1}.

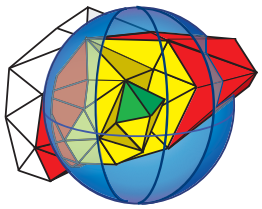


FIG. 3.8: **Tétraèdres actifs au sein de la zone d'intérêt**. La boule d'intérêt $\mathcal{B}$ est délimitée par la sphère bleue. Au sein de celle-ci, on distingue trois types de tétraèdres actifs : le tétraèdre *source* en vert d'où commence l'expansion via un algorithme de *croissance de régions* ; les tétraèdres actifs *internes* en jaune strictement inclus dans la zone d'intérêt ; et les tétraèdres actifs *de bord* en rouge qui intersectent la sphère.

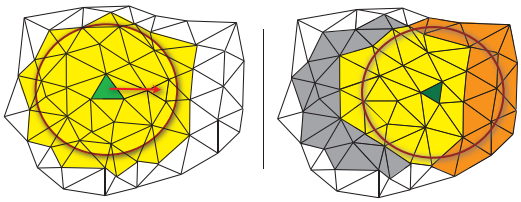


FIG. 3.9: **Mise à jour de la zone d'intérêt en utilisant la cohérence temporelle**. Pour plus de clarté l'illustration est réalisée sur des maillages triangulaires sans aucune perte de généralité. À gauche : le centre d'intérêt o de la boule $\mathcal{B}$ est déplacé vers la droite, résultant au déplacement du triangle *source* vert. À droite : mise à jour de la zone d'intérêt en utilisant la cohérence temporelle. Le triangle vert représente la nouvelle source. Les triangles jaunes ont été conservés, les gris ont été enlevés, les oranges ajoutés.

Pour répondre à cette constatation, on introduit la notion de *cohérence temporelle* qui repose sur le principe que deux maillages Σ_b^t et Σ_b^{t+1} entre deux rendus consécutifs sont proches l'un de l'autre géométriquement et topologiquement. Cette cohérence temporelle va être exploitée lors de la mise à jour de la zone d'intérêt afin d'assurer une meilleure complexité temporelle.

$\mathcal{B}_t$ a été déplacée en $\mathcal{B}_{t+1}$. La mise-à-jour est réalisée en deux temps (comme en section 2.1) : la détection du nouveau centre d'intérêt et la mise-à-jour de la zone d'intérêt en utilisant la cohérence temporelle.

Localisation du nouveau centre d'intérêt La première nécessité est de trouver le nouveau tétraèdre source s_{t+1}. On utilise la cohérence temporelle pour modifier la recherche au sein de l'octree. La recherche de la nouvelle source débute de la feuille de l'octree contenant l'ancienne source s_t. Si cette feuille ne contient pas o, alors l'octree est traversé en remontant aux parents jusqu'à ce que l'un d'eux la contienne. En pratique, la recherche ne remonte que de quelques niveaux pour des mouvements non brusques correspondant au cadre d'une exploration des résultats. Cela permet d'éviter le parcours de la profondeur totale de l'octree.

Mise-à-jour des tétraèdres inclus dans la nouvelle boule d'intérêt Une fois la racine s_{t+1} trouvée, les tétraèdres contenus dans la nouvelle zone d'intérêt $\mathcal{B}_{t+1}$ doivent être extraits. Deux cas sont alors possibles : $s_{t+1} \notin \mathcal{B}_t$ ou $s_{t+1} \in \mathcal{B}_t$.

Dans le premier cas, la table de hachage et la liste de bord remplies lors de la précédente extraction sont libérées et les tétraèdres actifs inclus dans $\mathcal{B}_{t+1}$ sont extraits sans cohérence temporelle, comme expliqué au sein de la section précédente 2.1.

Dans le second cas où $s_{t+1} \in \mathcal{B}_t$, les tétraèdres actifs sont extraits en trois étapes en prenant en considération les précédents tétraèdres actifs comme illustré en figure 3.9 :

1. **Mise-à-jour de la liste de bord.** La liste de bord contenant les tétraèdres actifs de bord est parcourue. Tout tétraèdre n'appartenant pas à la nouvelle zone $\mathcal{B}_{t+1}$ est enlevé et est stocké dans une nouvelle liste dite *extérieure*.

2. **Suppression de tous les tétraèdres n'appartenant pas à $\mathcal{B}_{t+1}$.** La liste extérieure (créée lors de l'étape précédente) est utilisée comme source de la recherche. En utilisant les relations d'adjacence, l'ensemble des tétraèdres actifs internes à $\mathcal{B}_t$ ne l'étant plus pour $\mathcal{B}_{t+1}$ sont enlevés de la table de hachage. De plus, tous les tétraèdres détectés comme actifs de bord sont ajoutés à la liste de bord.

3. **Ajout des nouveaux tétraèdres.** La liste de bord, fraîchement mise à jour, est utilisée comme point de départ. Une nouvelle expansion – par croissance de régions – est réalisée grâce aux relations d'adjacence. Les tétraèdres internes sont stockés dans la table de hachage, ceux de bord dans la liste.

2.3 Extraction du maillage birésolution

L'extraction globale du maillage birésolution est ainsi réalisée en trois temps :

Extraction des tétraèdres fins. L'ensemble des tétraèdres fins se trouvant au sein de la zone d'intérêt sont extraits comme expliqué au sein des sections 2.1 et 2.2 en utilisant si possible la cohérence temporelle. La liste des clusters actifs est mise à jour lors de cette étape.

Extraction du maillage grossier . Le maillage grossier est extrait en fonction des clusters inactifs. Afin d'éviter le parcours de l'ensemble des tétraèdres fins inactifs, le maillage grossier est calculé une unique fois au début de l'application dans sa globalité. Au cours de l'exploration, celui-ci est adapté en fonction du statut d'activité des clusters. Les tétraèdres grossiers ayant au moins un sommet dont le cluster image par la partition est actif ne sont pas extraits.

Transformation des tétraèdres de lien . L'ensemble des tétraèdres fins issu de la première étape est étendu à l'ensemble des tétraèdres actifs et de lien correspondant aux clusters actifs détectés. Pour cela, une liste de tétraèdres fins est construite pour chaque cluster lors du chargement des données (les ensembles τ_c définis en équation (2.1)) indexée selon les sommets grossiers. Au cours du parcours de cette liste, les tétraèdres de lien subissent une transformation en fonction de la surjection $\wp$ comme détaillé au sein de l'algorithme 2.

À la fin de ces trois étapes, un unique maillage birésolution est ainsi extrait.

2.4 Complexité et Résultats

Nous évoquons la complexité théorique en pire cas de nos algorithmes d'extraction et de mise-à-jour (section 2.4.1) que nous confrontons aux valeurs réelles d'implantation (section 2.4.2). Nous abordons ensuite les limitations d'une telle approche sur un processeur central en dernière section (section 2.4.3).

2.4.1 Complexité en pire cas

On considère indépendamment les trois étapes de la construction du maillage birésolution : l'extraction des tétraèdres actifs au sein de la boule d'intérêt, du maillage grossier et l'extension à la totalité des tétraèdres actifs et de lien.

Extraction des tétraèdres actifs au sein de la boule d'intérêt La complexité de l'extraction de la zone d'intérêt contenue dans la boule d'intérêt $\mathcal{B}$ est en $O(\mathcal{R})$ dans toutes les situations avec $\mathcal{R}$ représentant les tétraèdres inclus dans $\mathcal{B}$. On a $\mathcal{R} \leq \mathcal{F}$.

En effet, si l'on suppose que la profondeur maximale de l'octree est p, la complexité de l'extraction du tétraèdre source est dans le pire cas en $O(\log_2 p)$. La construction de la région via les relations d'adjacence est en $O(\mathcal{R})$ puisque l'insertion au sein d'une table de hachage et d'une liste est en $O(1)$.

Lors de la mise-à-jour, la complexité dans le pire cas pour la recherche au sein de l'octree reste inchangée. L'utilisation de la cohérence temporelle garantit que le coût de la mise-à-jour de la zone est égal au nombre de tétraèdres enlevés puis ajoutés au cours du déplacement. La table de hachage garantit une suppression en $O(1)$. Le parcours obligatoire des listes de bord et extérieur impose un coût dans le pire cas en $\alpha(\mathcal{R})$. Ce pire cas est équivalent à enlever tous les tétraèdres de la région d'intérêt.

L'extraction de la région à haute résolution au sein de $\mathcal{B}$ est donc dans le pire cas en $O(\mathcal{R})$ puisque la profondeur $p << \mathcal{R}$ rendant la recherche au sein de l'octree négligeable devant l'extraction.

Extraction du maillage grossier L'extraction du maillage grossier est en $O(\mathcal{G})$. Le maillage grossier est généré lors d'une phase de précalculs. Il est donc directement accessible en mémoire. Le test de tous les sommets afin de déterminer le statut de leur cluster image par la surjection $\wp$ entraîne le parcours de tous les tétraèdres grossiers.

Interne + Bord	Bord	Complexité
610	268	162
4450	898	758
22940	2750	3922
169250	10633	16060

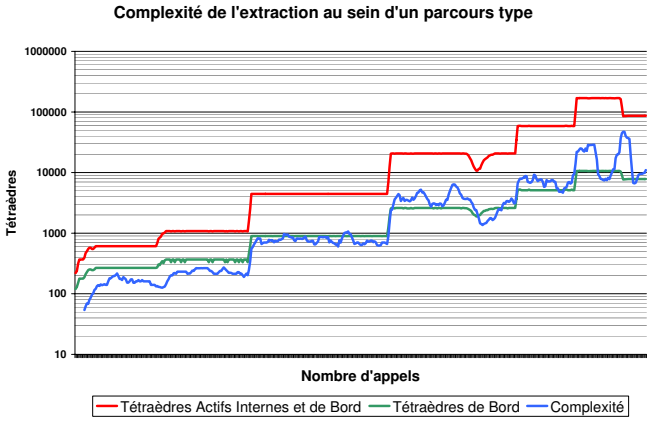


FIG. 3.10: **Complexité de la mise-à-jour de la zone d'extraction fine au sein d'un parcours-type dans un ensemble de données.** Les courbes représentent respectivement le nombre de tétraèdres actifs internes et de bord (en rouge), uniquement de bord (en vert) et la complexité réelle de la mise-à-jour (en bleu). À gauche, au sein du tableau, quelques valeurs clefs ont été extraites. Ces valeurs sont constatées et non empiriques.

Extension à l'ensemble des tétraèdres fins actifs et de lien L'extension à l'ensemble des tétraèdres fins actifs et de lien coûte uniquement la différence $O(\mathcal{F})$ grâce à l'utilisation de listes de tétraèdres associées à chaque cluster.

Complexité totale La complexité Γ_b de l'extraction du maillage grossier est donc égale à :

$$\Gamma_b = O(\mathcal{R} + \mathcal{G} + \mathcal{F}) \sim O(\mathcal{F} + \mathcal{G})$$

2.4.2 Résultats

Les temps d'extraction obtenus grâce à ce schéma reposant sur la cohérence temporelle sont fournis au sein du tableau 3.1. On remarque que le pire cas dont la complexité est en $O(\mathcal{R})$ (section 2.4.1) n'est ainsi jamais atteint. Il est en moyenne inférieur, représentant 18 % de $\mathcal{R}$.

La complexité réelle de l'algorithme d'extraction avec cohérence temporelle tend à être équivalente au nombre de tétraèdres actifs de bord de la zone d'intérêt courante comme illustré au sein de la figure 3.10 lors d'une exploration-type réalisée au sein de l'ensemble *BuckyBall* représentée en figure 3.11. Dans cette situation, en effet, le coût de la mise-à-jour représente en moyenne 109 % du nombre des tétraèdres de bord de la boule d'intérêt $\mathcal{B}$. Cette constatation semble pertinente, puisque dans le cadre d'une exploration, le mouvement a peu d'envergure au sein des données : cela revient à ne modifier que les tétraèdres de bord.

Ainsi on obtient en moyenne un taux d'extraction des tétraèdres fins de 788 000 à la seconde contre seulement 250 000 sans utiliser la cohérence temporelle. Si nous réalisons des comparaisons avec les précédentes approches et leur taux d'extraction fourni au sein du tableau 2.3, nous constatons deux choses.

La première est que le temps d'extraction des tétraèdres sans la cohérence temporelle est légèrement inférieur à celui de la création d'un niveau de détails issu d'un schéma multirésolution construits *via* des forêts d'arbres binaires [CDFM+04] ou une hiérarchie de segments avec décompression [SS06]. Néanmoins, elle reste plus performante que l'extraction réalisée dans la structure de données optimisée de forêts d'arbres binaires proposée par [SS05].

La seconde est que l'utilisation de la cohérence temporelle permet d'obtenir des taux d'extraction jamais atteints par les schémas multirésolution actuels puisque notre taux d'extraction de 778 mille tétraèdres à la seconde est supérieur à la hiérarchie de segments sans compression [SS06] qui était jusqu'à lors la meilleure avec un taux de 500 mille tétraèdres à la seconde.

La taille maximale des maillages pouvant être traités par cette méthode au sein du processeur central avec deux gigaoctets de mémoire vive est de l'ordre de 5 millions de tétraèdres puisque l'utilisation de la cohérence temporelle et d'un algorithme de croissance de régions imposent la connaissance des relations d'adjacence

Données	Tétraèdres fins	Tétraèdres extraits	Temps (ms)
Blunt Fin	222 414	3 050	4
		21 575	29
		134 680	188
Post	616 050	2 785	3
		35 960	44
		379 650	470
BuckyBall	1 250 235	32 320	41
		145 530	191
		439 860	456
Fuel	1 250 235	41 680	52
		308 800	389
		1 028 680	1 310
DTI	4 596 765	48 085	61
		328 660	432
		1 548 790	1 976

TAB. 3.1: **Temps d'extraction de la zone d'intérêt avec cohérence temporelle pour différents ensembles de données.** Les ensembles *Fuel* et *DTI* sont des grilles régulières tétraédralisées. Le nombre de sommets fins initial est indiqué. Plusieurs tailles de régions ont été imposées afin de faire varier le nombre de tétraèdres fins extraits. Les temps d'extraction avec utilisation de la cohérence temporelle sont en millisecondes.

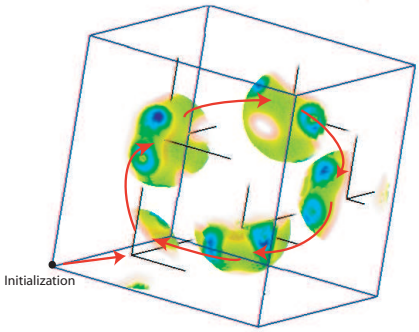


FIG. 3.11: **Exemple d'exploration au sein de l'ensemble de données *BuckyBall*.** La boule d'intérêt $\mathcal{B}$ est déplacée au sein de la simulation afin d'exploiter les résultats. À l'initialisation, celle-ci est placée dans un coin de la boîte englobante des données. Ensuite, l'utilisateur déplace $\mathcal{B}$ afin de mettre en évidence la structure de cage fermée des fullerènes (rappelant celle d'un *ballon de football*). La molécule C_{60} est ainsi composée de 60 carbones disposés aux sommets d'un polyèdre régulier de 0,7 nm de diamètre et dont les facettes sont 20 hexagones et 12 pentagones [KHO+85]. Le champ scalaire représente ici la densité de ces atomes de carbone.

entre cellules. C'est deux fois moins que pour la hiérarchie de segments avec compression mais supérieur aux autres précédentes méthodes.

Le couplage de cette extraction avec des techniques de rendu sera détaillé plus précisément au sein du chapitre 4 suivant.

2.4.3 Limitations

La première limitation de ce schéma est sa consommation mémoire. En effet, l'utilisation des relations d'adjacence double quasiment l'espace mémoire occupé par le maillage fin. Malgré l'augmentation continuelle de la mémoire vive des ordinateurs de bureau, une approche en mémoire externe de cette technique permettrait de résoudre ce problème. Une telle implantation est proposée pour l'extraction réalisée sur carte graphique au sein de la section 4.

La seconde limitation reste l'interactivité. En effet, la définition de la boule d'intérêt influence le nombre de tétraèdres fins extraits en son sein. Plus le rayon R de la boule $\mathcal{B}(o,R)$ sera grand, plus le nombre de tétraèdres fins $\mathcal{F}$ le sera. Or, le taux d'extraction, bien que supérieur aux précédentes techniques, ne va pas permettre du temps-réel mais plutôt de l'interactivité pour un grand nombre extrait de tétraèdres fins (au sens défini au chapitre 1, section 3.2). Une exploration non fluide des données pourrait perturber l'exploitation de celles-ci. Néanmoins, la localité des phénomènes intéressants comme exposé au sein du chapitre 1 ; permet d'induire que la zone d'intérêt ne représentera qu'un petit pourcentage des données et ainsi que l'exploration restera temps-réel (dans le pire cas interactive) même pour des maillages tétraédriques ayant plusieurs millions de cellules.

Une autre réponse algorithmique à cette limitation est d'utiliser la puissance de calculs et le parallélisme des cartes graphiques. Nous abordons ce point au sein de la section 3 suivante.

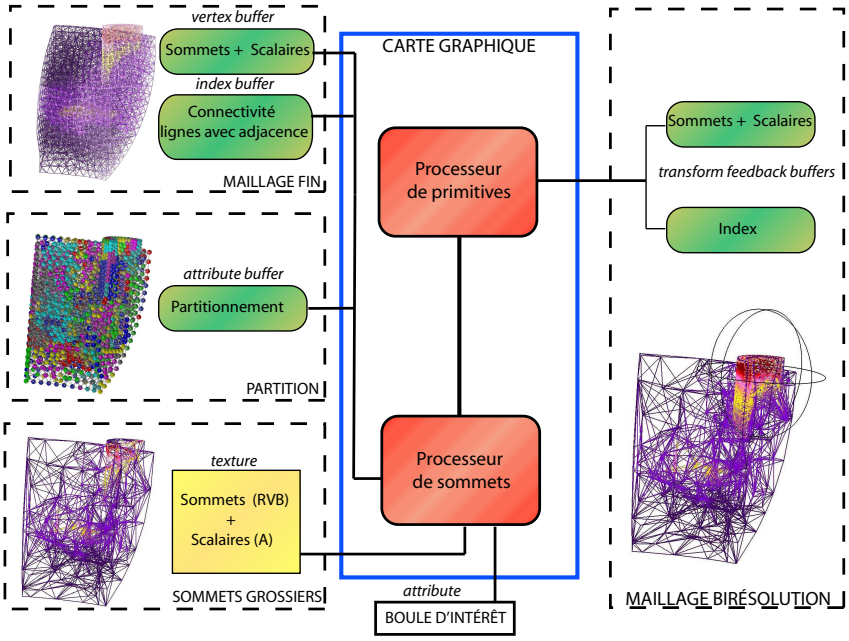


FIG. 3.12: **Extraction du maillage birésolution sur carte graphique**. Elle est illustrée sur l'ensemble de données *SPX* représenté en filaire. L'extraction prend en entrée : le maillage fin sous la forme de positions géométriques et de champ scalaire (*vertex buffer*) et de connectivité (*index buffer*), le partitionnement des sommets fins (*atrribute buffer*) et les sommets grossiers avec leur champ scalaire associé (*texture*). La carte graphique produit grâce à la connaissance de la boule d'intérêt en sortie du processeur de primitives le maillage birésolution. Celui-ci est encodé avec deux tampons (*transform feedback buffers*) : un contenant la géométrie et le champ scalaire, l'autre les indices des sommets. Plus de détails sont donnés au sein de la section 3.1.

3 Implantation sur la carte graphique

Avec l'avènement des nouvelles cartes graphiques[1], l'extraction optimisée sur processeur central (CPU) du maillage birésolution s'est révélée inefficiente. En effet, la programmation du processeur de primitives via un *geometry shader* a permis le portage de notre algorithme au cœur du pipeline de la carte graphique (cf. chapitre 1, section 4.2). Ce processeur de primitives permet de créer ou d'enlever de l'information géométrique et se révèle ainsi parfaitement adapté à l'extraction à la volée de géométrie. Au vu de la puissance de calcul des cartes graphiques (GPU), une telle implantation ne peut par conséquence qu'améliorer les taux d'extraction à condition d'adapter l'algorithme à l'architecture parallélisée des cartes graphiques.

Nous détaillons dans la suite de la section, l'implantation optimisée sur carte graphique de l'algorithme d'extraction (section 3.1) ainsi que les résultats et les limitations d'une telle approche (section 3.2).

3.1 Mise en œuvre

Le GPU représente une grande puissance de calcul au sein de laquelle les structures de données complexes sont difficilement transposables. Ainsi, les optimisations réalisées au sein du CPU en considérant la cohérence temporelle se révèlent inefficaces et difficilement transposables au sein du GPU. Celui-ci à l'inverse peut recalculer l'extraction à chaque mise-à-jour à moindre coût. L'algorithme 2 (section 1.2.2) a donc été directement transposé au sein de la carte graphique de la manière suivante.

À l'initialisation, les positions géométriques, le champ scalaire et le partitionnement des sommets fins sont envoyés de l'application au GPU *via* des tampons de sommets et d'attributs. La connectivité est transmise en

[1] du type *NVIDIA Ge80* et suivantes.

utilisant un tampon d'indices. Les tétraèdres n'étant pas des primitives fournies par les *API* standards (cf. chapitre 1, section 4.2), ceux-ci sont représentés de manière unique grâce à des "lignes avec adjacence"[2] pouvant encoder quatre sommets pour une primitive. Notons que les quadrangles ne peuvent être utilisés car ceux-ci vont être triangulés au sein de la carte graphique.

Les sommets grossiers sont eux-aussi envoyés une et une seule fois à l'initialisation sous la forme d'une texture bidimensionnelle $RVBA$ où les canaux RVB encodent la position dans l'espace, et A la valeur du champ scalaire.

Une fois ces informations contenues dans la carte graphique, le processeur de sommets est exécuté lors de l'exploration. Celui-ci transmet au processeur de primitives pour chaque sommet fin, le sommet grossier associé via le partitionnement ainsi que des informations supplémentaires au bon déroulement de l'extraction : le statut actif/inactif du sommet, le champ scalaire et un indice de tableau afin de pouvoir créer une soupe de polygones. Ces opérations sont réalisées par sommet dans un souci d'optimalité.

Les tétraèdres sont ensuite traités par le processeur de primitives. Les sommets fins de chaque tétraèdre sont examinés et modifiés en fonction de leur statut (actif ou non). Les sommets actifs sont conservés, les inactifs sont transformés par la surjection $\wp$ en leur image grossière. Cette dernière transformation permet de créer à la fois les tétraèdres de lien et les tétraèdres grossiers. Les tétraèdres dégénérés ne sont pas envoyés au tampon de sortie. De plus, chaque tétraèdre est représenté en sortie comme l'union de deux lignes afin de pouvoir le réutiliser comme des "lignes avec adjacences" lors de futurs traitements comme l'application d'une technique de visualisation (cf. chapitre 4).

Au cours de l'extraction, le tramage est désactivé afin d'empêcher la géométrie d'être transformée en pixels, puisqu'une telle transformation est inutile.

Deux tampons de sortie sont récupérés. Le premier contient une soupe de coordonnées de sommets groupées en quadruplets. Le second contient les indices des sommets originaux sous la forme de l'indice fin pour les sommets fins et du nombre de tétraèdres fins ajouté à l'indice grossier pour les sommets grossiers. Ces deux tampons représentent ainsi une soupe de cellules.

L'extraction au sein de la carte graphique permet ainsi d'obtenir en une seule étape le maillage birésolution. Cette étape est résumée au sein de la figure 3.12.

3.2 Résultats et Limitations

Les résultats présentés ont été réalisés avec une *GeForce 8800 GT / PCI / SSE2* possédant 512 mégaoctets de mémoire.

3.2.1 Temps d'extraction

Des temps d'extraction pour cette implantation au sein de la carte graphique sont donnés au sein de la figure 3.13. Ils ont été calculés puis moyennés à la suite d'un certain nombre d'exploration au sein du maillage *Fighter (r)*.

Le nombre moyen de tétraèdres extraits à la seconde est de neuf millions de tétraèdres. Ce résultat est à nuancer en fonction de l'ensemble de données à haute résolution initial. En effet, le temps d'extraction ne peut être inférieur à 170 ms pour cet ensemble de données. Pour un nombre peu élevé de tétraèdres fins extraits, ce coût obligatoire fait que l'algorithme est plus lent qu'une implantation optimisée pour le CPU. À l'inverse, le taux d'extraction croît rapidement dès que le nombre de tétraèdres extraits dépasse le million. Ainsi, l'utilisation de la carte graphique se révèle efficiente lorsque le maillage birésolution est constitué d'un nombre non négligeable de cellules, ce qui est l'objectif premier de cette implantation.

La complexité de notre algorithme sur carte graphique est donc une somme de deux composants qu'il faut analyser : une constante d'extraction γ et un surcoût éventuel ε dû à la transformation géométrique du maillage fin vers le maillage birésolution.

La constante d'extraction γ s'explique par le fait que la carte graphique parcourt et analyse l'ensemble du maillage fin. Elle correspond donc au traitement par les processeurs de sommets et de primitives des tétraèdres

[2] du type *GL_LINES_ADJACENCY_EXT*

Tétraèdres	Temps (ms)	Extraction (tet/s)
200 000	171,9	1 170 775
500 000	171,9	2 926 568
1 000 000	182,3	5 572 498
2 500 000	229,2	10 896 829
4 000 000	312,5	12 831 616
5 929 090	437,5	13 552 205

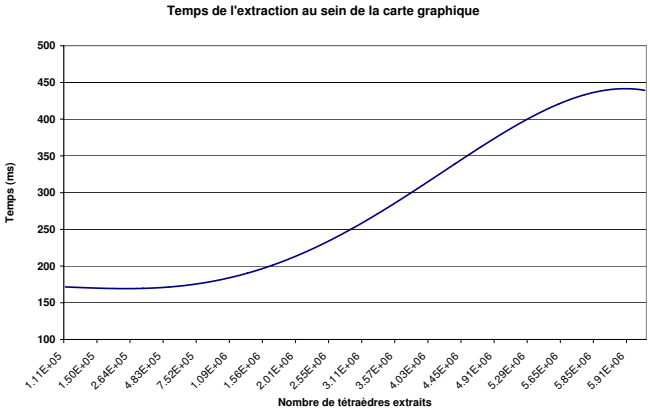


FIG. 3.13: **Temps d'extraction du maillage birésolution sur la carte graphique pour *Fighter (r)*.** La courbe bleue représente le temps d'extraction du maillage birésolution en fonction de son nombre de tétraèdres courant. Elle est une moyenne de plusieurs explorations au sein du maillage *Fighter (r)* composé de 5 929 090 tétraèdres fins. Quelques valeurs clefs avec le nombre des tétraèdres extraits à la seconde sont fournies dans le tableau de gauche.

fins initiaux. Cette constante a été évaluée afin de connaître la complexité de base de notre implantation sur la carte graphique.

Nous avons effectué les mêmes tests sur des maillages ayant un million de tétraèdres comme le *Torso* (1 082 723 tétraèdres) ou le *BuckyBall*. Les temps d'extraction sont constants et égaux à 31*ms*. Cela signifie que la carte graphique est capable de traiter en moyenne près de 35 000 000 tétraèdres en une seconde. Les temps d'extraction calculés pour des maillages encore plus petits (*SPX*, *BluntFin*) sont identiques.

La constante d'extraction est donc proportionnelle au nombre de tétraèdres fins initiaux et impose une limite inférieure à la complexité de l'extraction. D'après nos expériences γ peut être évaluée à : $\gamma \approx \frac{card\ T_f}{33\ 000}$.

En plus de cette constante, un surcoût ε intervient dû à la transformation géométrique du maillage initial en maillage birésolution : création des tétraèdres grossiers et de lien. Ce surcoût peut être négligeable en-dessous du million de tétraèdres fins (comme pour le *Torso*) ou augmenter lentement si le nombre de tétraèdres à extraire au sein du maillage birésolution est plus grand que le million (comme pour le *Fighter (r)* dans la figure 3.13).

Pour conclure, la rapidité de notre algorithme d'extraction est ainsi fortement dépendante comme nous l'attendions au nombre de tétraèdres fins composant le maillage initial avant toute extraction. Néanmoins, la puissance de calcul de la carte graphique permet de compenser ce surcoût et de dépasser sans équivoque la version CPU de notre algorithme pour des gros maillages de données. En effet, pour plus de 5 millions de tétraèdres fins initiaux, la version CPU n'extrait en moyenne que 778 milles tétraèdres à la seconde contre 9 millions sur la version GPU.

3.2.2 Limitations

L'implantation de l'extraction sur la carte graphique a permis d'accélérer fortement cette phase. Elle a aussi permis de diminuer l'espace mémoire utilisé à son exécution puisque les relations d'adjacence précédemment nécessaires pour déterminer et mettre à jour la région fine sont superflues. La connectivité du maillage grossier n'est plus précalculée afin d'optimiser les performances et ainsi seuls les sommets sont stockés au sein de la mémoire de la carte graphique. L'espace mémoire nécessaire pour un maillage est donc fortement diminué.

Néanmoins, la taille de la mémoire d'une carte graphique est souvent restreinte par rapport à celle du processeur principal d'un ordinateur. La place ainsi économisée permet en pratique de traiter les même ensembles de données qu'avec la version CPU. De plus, la carte graphique impose une limite *hardware* non négligeable sur la taille des tampons de communication qui est à prendre en compte lors de la génération du maillage unique. Pour ces raisons, et aussi afin de diminuer le coût de la constante γ d'extraction, une solution en mémoire externe a été développée. Nous la détaillons au sein de la section 4 suivante.

Signalons enfin une des aberrations des *API* actuelles. La création d'un tampon de données pour l'envoi à la carte graphique provoque lors de l'utilisation d'*OpenGL* et du langage de programmation *GLSL* une copie de sauvegarde de ce tampon au sein de la mémoire du processeur afin que l'information reste disponible en cas de défaillance de la carte graphique. Le CPU ne voit donc en aucun cas son occupation mémoire diminuer lors des envois. Cette constatation a aussi grandement motivé la création d'une application en mémoire externe. Notons, tout de même que de tels comportements n'ont pas été remarqués avec l'utilisation du langage *CUDA*.

4 Implantation en mémoire externe

Au sein de cette section, nous présentons une version dite en mémoire externe (ou *out-of-core*) de l'extraction de notre maillage birésolution permettant le traitement par la carte graphique de maillages tétraédriques composés de plus de dix millions de cellules. Elle est, à notre connaissance, la première méthode *out-of-core* permettant de réaliser l'extraction d'un maillage tétraédrique non monorésolution. En effet, les précédentes approches reposant à la fois sur des algorithmes de simplification locale et des structures de données complexes comme les forêts d'arbres binaires ou les hiérarchies de segments sont difficilement transposables pour traiter de gros ensembles de données en mémoire externe.

A contrario, notre méthode simple reposant uniquement sur un partitionnement des sommets fins, permettant ainsi un traitement complet sur carte graphique, est idéale pour être transposée en mémoire externe *via* de légères modifications des prétraitements et des calculs que nous détaillons ci-après.

Dans la suite, nous revenons dans un premier temps sur les précédentes structures de données utilisées pour construire des méthodes *out-of-core* (section 4.1) pour les maillages surfaciques et en visualisation scientifique. Puis nous décrivons l'architecture choisie (section 4.2) avant d'évaluer et de discuter les résultats (section 4.3).

4.1 Principe des méthodes en mémoire externe

Face à l'augmentation constante des masses de données, comme évoquée au sein du chapitre 1, section 5.1, celles-ci ne sont plus chargeables entièrement au sein de la mémoire centrale d'un ordinateur de bureau. Une solution à ce problème majeur est de développer des méthodes en mémoire externe permettant de charger par morceaux les ensembles de données. Nous évoquons brièvement au sein de cette section les différentes structures de données qui ont été développées pour les maillages irréguliers triangulés et tétraédriques afin de réaliser cette lecture en mémoire externe. Le lecteur intéressé par plus de détails peut se référer à [SCESL02].

Pour réaliser une approche *out-of-core*, l'ensemble de données (géométrie et connectivité) doit être découpé en *blocs* pouvant être chargés en mémoire centrale. Un bloc contient ainsi un nombre limité de coordonnées de sommets ou de tétraèdres. La création et le contenu de ces blocs (on parle aussi de *pagination*) sont les enjeux de ces méthodes.

Une premier prétraitement possible est, lorsque le maillage est stocké au sein d'une structure indexée (cf. chapitre 1, section 2.3), d'effectuer en premier lieu un *déréférencement* des pointeurs vers les indices de sommets [CS97]. On rappelle qu'au sein de cette structure, chaque tétraèdre est encodé par quatre pointeurs, un sur chacun de ses sommets. Lors de la lecture d'un tétraèdre, ces pointeurs peuvent provoquer le chargement au sein de la mémoire centrale de quatre blocs de coordonnées de sommets (un par sommet) dans le pire cas. Cela peut ainsi provoquer de multiples opérations d'entrées-sorties (E/S) coûteuses et une saturation de la mémoire vive. Pour éviter ces problèmes, le déréférencement remplace les pointeurs des sommets par leurs valeurs, c'est-à-dire leurs coordonnées tridimensionnelles. Il est réalisé dans sa globalité en mémoire externe.

Pour découper le maillage tétraédrique en blocs, de nombreuses solutions existent reposant sur le plongement du maillage au sein de structures de données adaptées :

- grille régulière [Lin00, NNSM07] utilisée pour la simplification surfacique de maillages ;
- octree [CMRS03] utilisé aussi pour la simplification surfacique de maillages ;
- BSP-Tree [GM08] utilisé pour le rendu d'isosurfaces et de maillages triangulés ;
- hiérarchie de tétraèdres [CGG+04] utilisée pour la simplification surfacique de maillages ;
- ou encore des *métacellules* (*metacells*) [CSS98] permettant de créer des regroupements de tétraèdres de taille fixe, utilisées pour l'extraction d'isosurface et le rendu volumique direct.

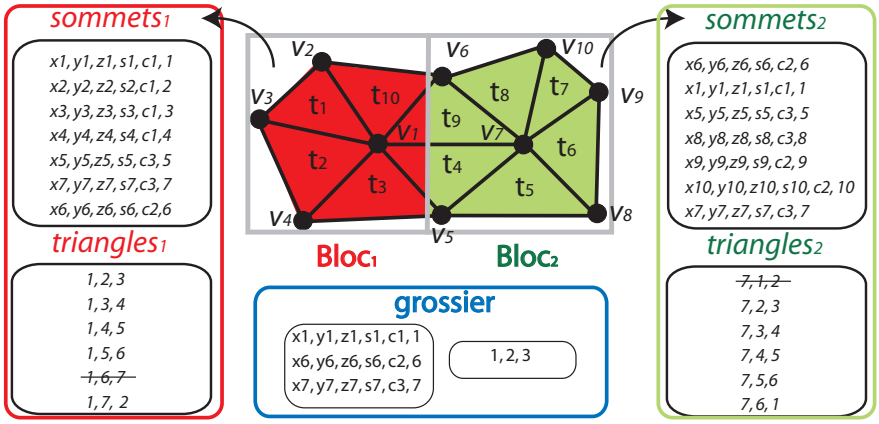


FIG. 3.14: **Pagination utilisée pour notre implantation en mémoire externe.** Le maillage fin est composée de 10 sommets v_i et 10 triangles t_i. Il est associé à un maillage grossier composé d'un triangle à trois sommets c_1, c_2, c_3 (non représenté). Ce maillage est plongé dans une grille régulière de taille $2 \times 1 \times 1$, générant deux blocs (respectivement en vert et en rouge). Les fichiers générés de sommets et de triangles sont renseignés pour chaque bloc. Pour le fichier de sommets contenant les coordonnées x_i, y_i, z_i, l'attribut scalaire s_i, le sommet grossier associé c_j et l'indice i original, les indices utilisés sont les initiaux. Pour le fichier de triangle, le pointeur (ici un entier) est indexé selon le fichier de sommets associé. Enfin, une structure indexée de triangles est créée à part pour l'extraction du maillage grossier (en bleu). Les triangles stockés dans ce fichier sont retirés des précédents fichiers (barrés dans les fichiers correspondants).

Récemment, une autre solution a été proposée afin de remplacer ce découpage en blocs via la création d'une nouvelle représentation des maillages : les *streaming meshes* [IL05, VCL+07]. Cette représentation regroupe localement les cellules et les sommets au sein du fichier afin, lors de la lecture, d'allouer puis de désallouer la mémoire dès qu'un sommet devient inutile.

Un cache et le préchargement des données (*prefetch*) sont habituellement utilisées lors de l'exécution des algorithmes *out-of-core* afin de minimiser les opérations d'entrées/sorties et la latence que celles-ci pourraient entraîner.

4.2 Implantation

Au sein de cette partie, nous détaillons dans un premier temps la création des blocs (section 4.2.1) lors d'un prétraitement s'inscrivant dans la continuité de ceux réalisés au sein du chapitre 2, section 2. Puis, nous détaillons l'extraction du maillage birésolution adaptée à cette nouvelle structure (section 4.2.2).

4.2.1 Précalculs : création de la pagination

Nous avons développé notre propre solution en mémoire externe dans l'optique d'utiliser notre schéma birésolution sur des gros ensembles de données et répondre aux différentes limitations (mémoire, tailles des tampons d'envoi et de retour) de la carte graphique évoquées au sein de la section 3. Pour la création des blocs, nous supposons que la structure indexée, le champ d'attributs et le partitionnement des sommets fins sont entièrement chargeables au sein de la mémoire centrale de l'ordinateur. Si tel n'est pas le cas, d'autres solutions sont envisageables - cf. section 4.3.3. La pagination obtenue est illustrée au sein de la figure 3.14 pour un maillage triangulaire.

Le maillage tétraédrique est plongé au sein d'une grille régulière fixée par la boîte englobante du volume et dont les différentes résolutions (en x, y et z) sont définies par l'utilisateur. Chaque voxel de la grille régulière définit un *bloc*. Chaque bloc sera représenté par un tuple de fichiers.

Les tétraèdres fins sont traversés une première fois afin de déterminer leur emplacement au sein des blocs grâce à la localisation spatiale de leurs sommets. Si un tétraèdre appartient à plusieurs blocs alors celui-ci sera inscrit dans chacun des différents blocs. Les sommets sont de plus réindexés pour chaque bloc.

Un second parcours permet d'inscrire dans chaque fichier uniquement les sommets avec leur attributs et sommet grossier associés ainsi que les cellules appartenant à chacun des blocs. On obtient ainsi une structure indexée indépendante pour chacun des blocs.

Afin de pouvoir gérer le rendu volumique direct – cf chapitre 4 – un fichier supplémentaire est généré pour chaque structure indexée stockant pour chaque sommet son indice dans le maillage fin initial. Cela permettra ainsi de ne pas confondre au sein de la carte graphique deux sommets fins ayant le même indice mais n'appartenant pas au même bloc.

Cette pagination permet ainsi de retrouver l'ensemble du maillage fin. Mais elle n'est pas suffisante. En effet, comme vu au sein de la section 3, le maillage grossier est extrait à la volée à partir du maillage fin. Une telle solution n'est plus envisageable avec une approche *out-of-core* puisque le maillage fin ne sera jamais chargé entièrement en mémoire, provoquant une reconstruction incomplète du maillage grossier.

Pour répondre à cette problématique, les tétraèdres fins décelés comme appartenant à quatre clusters différents (au niveau de la partition des sommets fins) sont de la même manière stockés au sein d'une structure indexée permanente qui permettra au cours de l'exécution d'extraire le maillage grossier correspondant. Ces tétraèdres de plus ne sont pas inscrits dans les fichiers décrits précédents (ceux associés aux blocs) afin d'éviter les redondances.

4.2.2 Extraction dynamique

Cette pagination a été utilisée afin de réaliser une extraction du maillage birésolution pour des maillages ne pouvant être explorés jusqu'alors ni avec l'implantation CPU (section 2) ni GPU (section 3).

À l'initialisation, la structure indexée représentant le maillage grossier ainsi que celles stockant les blocs intersectés par la boule d'intérêt $\mathcal{B}$ sont chargés au sein de la mémoire centrale. La première est envoyée directement à la carte graphique une et une seule fois et n'est pas conservée en mémoire vive. Les secondes seront envoyées au fur et à mesure au cours de l'extraction.

Afin d'optimiser les performances, un cache au sein de la mémoire principale est utilisé. Un certain nombre de tampons mémoire sont créés à l'initialisation. Ce nombre est fixé en fonction de la taille maximale du plus grand bloc et de la taille de la mémoire centrale. Les structures indexées sont ainsi sauvegardées au sein de ces tampons.

Lors de l'exploration, seuls les blocs intersectant la boule d'intérêt $\mathcal{B}$ sont envoyés à la carte graphique. Les blocs intersectant pour la première fois $\mathcal{B}$ sont chargés et conservés au sein du cache tant que cela est possible. Dès que la mémoire est saturée, les tampons correspondant à des blocs inutiles sont écrasés par les nouveaux blocs intersectant $\mathcal{B}$. Ce système présente l'avantage d'utiliser la cohérence temporelle de l'exploration – comme nous l'avions déjà proposé en section 2.2. En effet, un bloc intersectant $\mathcal{B}_t$ sera avec une haute probabilité encore intersecté par $\mathcal{B}_{t+1}$ lors du rendu suivant. L'utilisation d'une file à priorité sur les tampons permet ainsi de conserver l'information et empêche des communications intempestives coûteuses avec le disque dur.

Il peut aussi arriver que la zone d'intérêt intersecte plus de blocs que la mémoire centrale ne peut en contenir. Dans ce cas, tout fichier supplémentaire est lu en écrasant le dernier tampon du cache et est directement envoyé à la carte graphique. Le fichier écrasé devra donc être relu pour chaque rendu successif si la situation persiste au cours de l'exploration. Naturellement, le nombre d'opérations d'entrées/sorties est dans ce cas élévé et entraîne un surcoût qui peut se révéler non négligeable.

Afin d'extraire le maillage birésolution et de contourner les limitations architecturales des cartes graphiques, seulement un triplet de tampons sommets-attributs-indices est créé pour l'envoi des données. Ce triplet permet ainsi d'envoyer successivement tous les blocs chargés en mémoire pour un rendu. Le même *shader* défini précédemment en section 3 est utilisé et le résultat de tous ces rendus consécutifs est stocké au sein d'un seul tampon de sortie géométrique. L'extraction au sein de la carte graphique est donc complètement indépendante de la solution utilisée (avec ou sans une approche en mémoire externe).

	Bucky	Fighter (r)	Spx (r)	Sf1	Engine
Tétraèdres fins	1,250,235	5,929,085	10,911,011	13,980,162	41,943,040
Taille de la grille	$2 \times 2 \times 2$	$4 \times 4 \times 4$	$4 \times 4 \times 4$	$3 \times 3 \times 3$	$4 \times 4 \times 4$
Partition Spatiale (s)	14	178	402	143	418

TAB. 3.2: **Temps de précalculs pour la construction de la structure en mémoire externe.** Temps en seconde pour le découpage des maillages tétraédriques en blocs indépendants afin de réaliser une partition spatiale. Cette étape est ajoutée aux deux étapes de précalculs détaillées au sein du chapitre 2 : la simplification (optionnelle) du maillage fin initial et le calcul du partitionnement des sommets fins. La taille de la grille régulière choisie pour les trois dimensions est précisée.

4.3 Résultats et Discussions

4.3.1 Calcul de la partition spatiale

La partition spatiale du maillage fin est réalisée lors de l'étape de précalculs et vient ainsi s'ajouter aux précédents prétraitements comme l'extraction optionnelle d'un maillage grossier et le calcul du partitionnement des sommets fins décrits au chapitre 2, section 2. Les temps globaux pour l'ensemble des précalculs sont disponibles au sein de l'annexe B.

Concernant la pagination du maillage fin en blocs, le tableau 3.2 contient les temps d'extraction pour un échantillon représentatif de maillages tétraédriques ainsi que la taille des grilles choisies pour les trois dimensions. La taille de la grille, le nombre de tétraèdres fins à traiter et parmi eux, les tétraèdres appartenant à plusieurs blocs, influencent les temps d'extraction. En effet, l'écriture de la partition spatiale au sein des différents fichiers est en $O(|T_f|)$ où $|T_f|$ est le nombre de tétraèdres fins. Néanmoins, chaque tétraèdre appartenant à plusieurs blocs sera écrit plusieurs fois. La régularité de la grille influence donc fortement la rapiditié du calcul. Ainsi, pour les ensembles de données *Bucky* et *Engine*, les temps d'extraction sont en $|T_f|$ puisque ces maillages tétraédriques proviennent à l'origine d'une grille régulière. Cela ne se vérifie pas avec les trois autres ensembles qui sont des maillages irréguliers natifs.

4.3.2 Extraction

Nous avons éprouvé notre implantation en mémoire externe sur l'ensemble des données fournies au sein du tableau 3.2 dont le maillage *Engine* possédant près de quarante-deux millions de tétraèdres.

Les temps d'extraction pour une exploration type au sein de l'ensemble *Sf1* sont fournis au sein du graphe de gauche de la figure 3.15. L'exploration est réalisée au sein d'une grille $2x2x2$. Deux courbes sont représentées. La première (rouge) indique le temps d'extraction au sein de la carte graphique, la seconde le temps des E/S pour le chargement des fichiers manquants. Le temps d'extraction réel est donc une somme des deux. Un histogramme est aussi fourni (en bleu) afin de mettre en évidence le nombre de fichiers maintenu au sein de la mémoire centrale. Nous avons limité ce nombre à huit ce qui permet de charger en mémoire l'ensemble du maillage fin. Rappelons que dans ce cas, les blocs sont envoyés un à un à la carte graphique ce qui évite les problèmes de débordement lors de l'allocation des tampons permettant de communiquer avec la carte graphique et de ne point saturer sa mémoire limitée.

En moyenne, le taux d'extraction contenant à la fois le chargement des fichiers et le calcul du maillage birésolution est compris entre trois à quatre millions de tétraèdres à la seconde. Ce taux est environ trois fois inférieur à la version entièrement implantée au sein de la carte graphique (section 3) mais elle permet de traiter des maillages qui ne pouvaient l'être auparavant. De plus, notre implantation assez naïve peut être améliorée sur de nombreux points comme nous le développons au sein de la section 4.3.3. Ce taux reste largement supérieur aux dernières techniques multirésolution tétraédriques [SS06].

Remarquons, de plus, que contrairement à la méthode précédente (proposée en section 3), seulement une partie des sommets fins est lue. La complexité de base de l'extraction est donc fortement réduite puisque celle-ci se limite à l'ensemble des tétraèdres fins effectivement envoyés à la carte graphique. Pour vérifier cette assertion, on réalise la même expérience que pour la méthode précédente sur l'ensemble de données *Fighter (r)* en supposant que l'ensemble des blocs est chargeable en mémoire (ici 64 pour une grille $4 \times 4 \times 4$). La constante d'extraction $\gamma = 170$ *ms* imposée par la méthode précédente n'existe plus puisque par exemple, pour 80 000 tétraèdres, l'extraction est réalisée en 31 *ms*, pour 200 000, 101 *ms* et pour 500 000, 148 *ms*. Néanmoins,

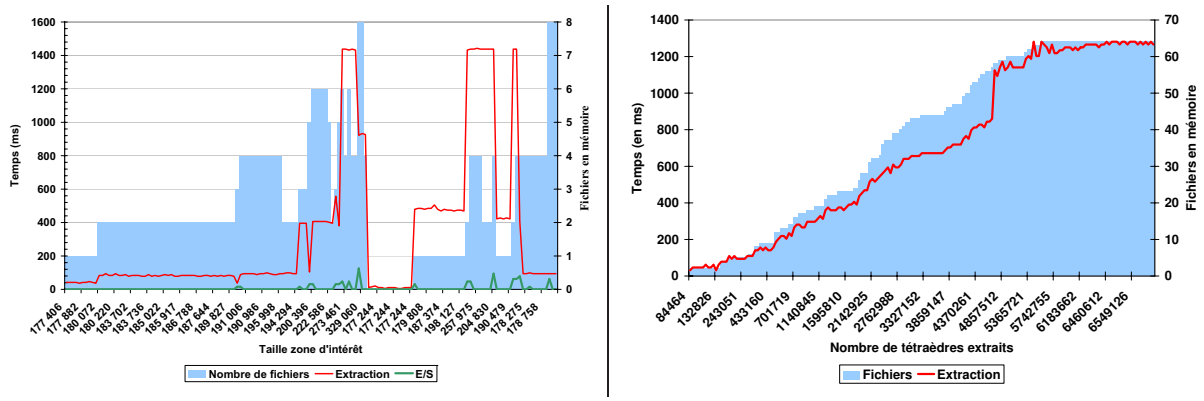


FIG. 3.15: **Temps d'extraction pour la méthode en mémoire externe au sein des ensembles *Sf1* et *Fighter (r)*.** À gauche, le nombre de tétraèdres extraits au cours d'un parcours type de *Sf1* est donné le long de l'axe des abcisses. Le temps d'extraction (en ms) du maillage birésolution au sein de la carte graphique est représenté par la courbe rouge. Le temps de lecture (en ms) des fichiers non encore chargés sur le disque est fourni *via* la courbe verte. Le temps réel d'extraction est donc une somme des deux courbes. Afin de mesurer l'occupation mémoire, l'histogramme bleu indique le nombre réel de fichiers conservés au sein de la mémoire lors de ce parcours. À droite, le temps d'extraction global pour *Fighter (r)* est donné avec le nombre de fichier maintenu en mémoire. À cause du découpage au sein de la grille régulière, le nombre de tétraèdres fins extraits à pleine résolution est supérieur au nombre initial.

au delà, à cause des lectures de fichiers et l'envoi successif des tampons, les performances sont diminuées par rapport à la version précédente, comme nous l'avons déjà souligné. L'ensemble de ces résultats est illustré par le graphe de droite de la figure 3.15.

Enfin, la création de répétitions de certains tétraèdres fins dans la structure de données en mémoire externe – ceux appartenant à plusieurs blocs et donc référencés dans autant de tuples de fichiers – entraîne les mêmes répétitions au sein du maillage extrait puisque la carte graphique, à cause du parallélisme au niveau des primitives, ne peut détecter de telles aberrations. Ces répétitions impliquent un surcoût (ici 6 572 394 au lieu de 5 929 090) mais ne nuit en aucun cas à la validité du maillage résultant et donc à la validité des images créées par les techniques de visualisation implantées en aval de l'extraction.

Notons que notre approche est la première à notre connaissance permettant l'extraction d'un maillage non monorésolution dont la résolution fine possède plus de quarante millions de tétraèdres au sein d'un ordinateur de bureau. Deux exemples de résultats générés grâce à notre schéma birésolution sont présentés au sein de la figure 3.16 : une extraction d'isosurface et un rendu volumique direct par lancer de rayons. L'intégration de telles techniques de visualisation à la suite de notre schéma birésolution afin de favoriser l'exploration interactive des données est détaillée dans le chapitre 4 suivant.

4.3.3 Discussion

Lors du calcul de la partition spatiale, il fut supposé que l'ensemble des données (structure indexée, champ d'attribut et partition des sommets fins) était chargeable au sein de la mémoire principale de l'ordinateur. Si ce n'est pas le cas, on peut étendre facilement notre partitionnement spatial en remplaçant la structure indexée par une soupe de cellules où chaque sommet est représenté par ses trois coordonnées spatiales (déréférencement), son champ d'attribut et son sommet grossier associé. Le partitionnement spatial pourra alors être réalisé via un tri externe des coordonnées successif selon les trois coordonnées (x, y et z) ce qui permettra de retrouver le partitionnement similaire à celui obtenu avec la grille régulière [CSS98].

Notre implantation en mémoire externe reste limitée. Quelques améliorations seraient envisageables afin d'augmenter les performances. Nous avons pour l'instant opté sur le fait que la lecture des fichiers de données était réalisée *via* le processeur central et que les informations placées dans le cache sont envoyées successivement à la carte graphique. Le dédoublement des tampons assurant la communication des données entre la mémoire centrale et la carte graphique pourrait être envisagé si la taille des blocs le permet (c'est-à-dire si les

deux tailles maximales de l'ensemble des blocs additionnées ne sont pas supérieures à la limite architecturale).

Un certain temps de latence est de plus causé par la lecture des nouveaux blocs au moment où la région d'intérêt les intersecte. Comme le processeur central est en attente lorsque la carte graphique travaille, celui-ci pourrait réaliser des prédictions (*prefetch*) sur les prochains blocs qui pourraient être intersectés et les charger en parallèle du rendu afin que l'ensemble des informations soit déjà au sein de la mémoire centrale pour le rendu suivant.

5 Bilan et Perspectives

Nous avons présenté au sein de ce chapitre plusieurs méthodes afin d'extraire un maillage birésolution guidé par la définition d'une zone d'intérêt. Cette extraction repose sur la connaissance d'un maillage fin et sur la définition d'un partitionnement des sommets fins contraint par un maillage grossier.

En premier lieu, une version implantable au sein de la mémoire centrale a été proposée. Afin d'en minimiser la complexité, les relations d'adjacence entre les cellules et la cohérence temporelle lors de l'exploration ont été exploitées. Malgré tout, l'occupation mémoire reste conséquente et les temps d'extraction, bien que supérieurs à l'état de l'art, ne sont pas encore assez rapides pour garantir une exploration temps-réel dans toutes les situations.

L'arrivée d'une nouvelle génération de cartes graphiques permettant la modification du traitement des primitives *via* un processeur dédié programmable a permis l'implantation du schéma d'extraction au sein de la carte graphique. Les temps d'extraction sont grandement améliorés mais les limitations architecturales des cartes graphiques ne permettent pas de traiter des maillages plus gros que la version en mémoire centrale.

Pour résoudre cette limitation, une première ébauche d'un traitement en mémoire externe a été développée permettant de traiter de gros ensembles de données. Les coûteuses opérations d'entrées/sorties ont été limitées grâce à l'utilisation d'un cache en mémoire centrale reposant sur la cohérence temporelle. L'extraction est toujours réalisée au sein de la carte graphique afin de conserver la puissance de calculs de celle-ci. Jusqu'à quarante millions de tétraèdres ont pu être ainsi traité de manière interactive grâce à ce schéma.

Le maillage birésolution ainsi extrait doit maintenant être exploité *via* des techniques de visualisation. Nous développons l'intégration de ces techniques au sein de notre schéma dans le chapitre 4 suivant.

Néanmoins, de nombreux travaux futurs restent encore envisageables et souhaitables. Outre l'amélioration de la technique en mémoire externe déjà évoquée précédemment, l'intégration de ce schéma au sein d'une architecture client/serveur semble une prochaine étape naturelle. En effet, seul le maillage grossier et les clusters actifs sont nécessaires à la construction du maillage birésolution. Ainsi, la résolution grossière pourrait être transmise à l'initialisation et les clusters actifs transmis au fur et à mesure de l'exploration en fonction des besoins. Un système de caches (sans doute disque dur, mémoire centrale et carte graphique) permettrait ainsi d'assurer un stockage sur trois niveaux architecturaux et de coupler une connexion réseau avec notre approche en mémoire externe.

L'intégration des nouvelles interfaces hommes/machines – comme les bras PHANTOM ou les Wiimotes – pour l'exploration permettrait d'ajouter une dimensionnalité supplémentaire à l'exploitation des données pour des utilisateurs experts. En effet, la définition d'une boule d'intérêt manipulable par l'utilisateur représente un atout non négligeable dans la compréhension de la localisation spatiale des informations importantes au sein des données. L'utilisation de tels outils permettraient ainsi une intégration de la profondeur, dimension manquante et difficilement mesurable lorsque la caméra est fixe [SP93], au cours de l'exploration. La simplicité et le côté "naturel" de ces interactions semblent aussi un atout majeur pour favoriser une compréhension rapide – et aussi une prise en main rapide – d'un tel outil par les utilisateurs.

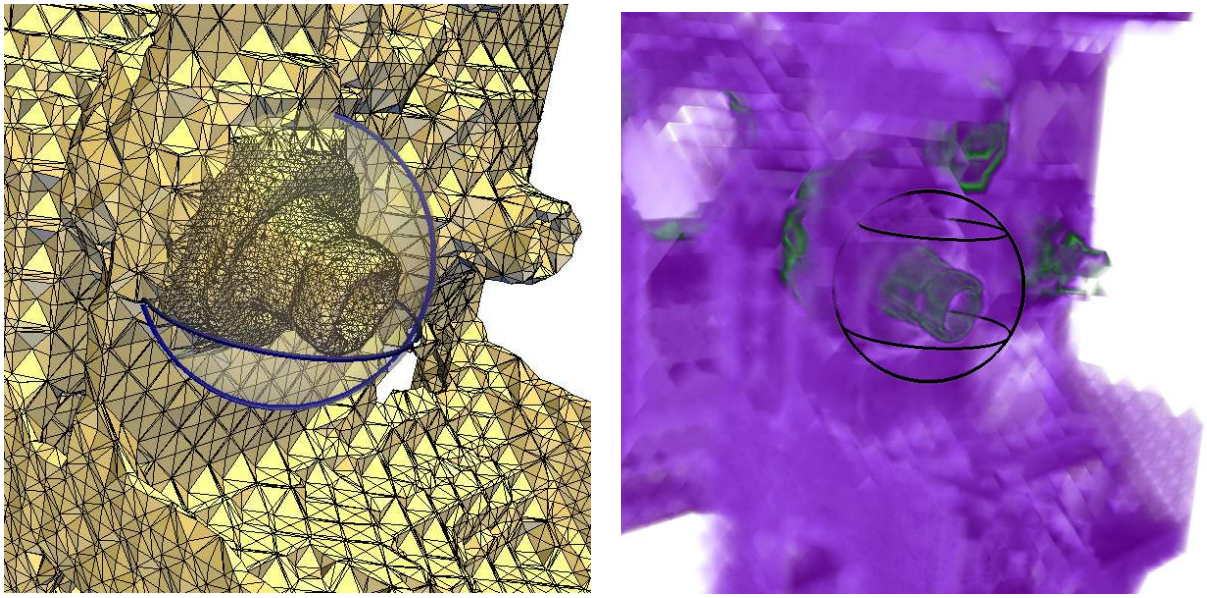


FIG. 3.16: **Maillage birésolution pour l'ensemble de données *Engine*.** Le maillage fin initial est composé de près de 42 millions de tétraèdre. Le maillage grossier est choisi afin de conserver l'interactivité tout en limitant la détérioration du champ scalaire. À gauche : extraction d'une isosurface. À droite : rendu volumique direct par lancer de rayons. La boule d'intérêt est représentée en filaire. En son sein, la résolution fine est extraite alors que la résolution grossière est conservée à l'extérieur. Aucun espace vide n'existe entre les deux résolutions.

CHAPITRE 4

Visualiser via le Rendu Volumique

> Scientific Visualization is the art of making the unseen visible, through the use of the 2D and 3D computer graphics coupled with high speed computations. [...] It can be used to understand and solve scientific problems, find hidden patterns in data and create models and visualizations for ideas that were previously conceptual.
>
> Clifford A. PICKOVER - *Frontiers of Scientific Visualization*, 1993

AFIN D'EXPLOITER les résultats issus d'une simulation, l'ingénieur ou le chercheur doit pouvoir les visualiser interactivement. La construction en amont de schémas multirésolution – comme notre approche birésolution – permet une exploration interactive de ces données *via* la mise-à-jour dynamique du nombre de cellules en fonction des besoins de l'utilisateur. Mais cela n'est pas suffisant. La compréhension d'un phénomène puis la communication des résultats obtenus à des tierces personnes, spécialistes ou non, passent principalement par le sens le plus utile à l'être humain : sa vue. La création d'une image porteuse de sens, d'une certaine qualité visuelle est ainsi une nécessité, en plus de la garantie de l'interactivité.

L'une des techniques de visualisation permettant de comprendre dans sa globalité un phénomène est le rendu volumique direct. Une telle technique assure l'affichage de l'ensemble du champ scalaire dans son domaine spatial afin de mettre en évidence ses variations, ses singularités, ses comportements inattendus ou inhabituels permettant de valider, comprendre ou réfuter une simulation. Contrairement aux techniques indirectes comme les plans de coupe ou l'extraction d'une isosurface, le rendu volumique direct garantit l'affichage de la structure globale du champ tout en assurant, grâce à la modulation de l'opacité, à la fois une meilleure visibilité de certaines valeurs d'intérêts mais aussi une gestion des problèmes perceptifs d'occlusion.

Le couplage d'une telle technique de visualisation avec des méthodes dites multirésolution semble ainsi une solution garantissant un traitement des données issues de simulations adapté aux besoin finaux de l'utilisateur. Ces besoins sont la génération d'une image porteuse de sens, facilement compréhensible, assurant la mise en évidence de zones d'intérêts, générée à des temps interactifs afin de faciliter l'exploitation des résultats issus

de simulation.

Au sein de cette section, nous détaillons ainsi plus précisément l'intégration du rendu volumique direct ordonné au sein du schéma birésolution proposé au sein du chapitre 3 précédent. Nous présentons, dans un premier temps, les précédentes approches permettant de réaliser un rendu volumique direct ordonné pour les grilles tétraédriques au sein de la section 1. Puis, nous détaillons l'intégration de deux algorithmes de rendu volumique existants que nous avons adaptés à la mise-à-jour dynamique des maillages tétraédriques : l'un pour l'extraction birésolution en mémoire centrale (section 2) ; l'autre pour l'extraction birésolution sur carte graphique (section 3). Nous détaillons enfin l'intégration des autres techniques de visualisation usuelles : iso-surfaces, plans de coupe, rendu par points ; au sein de la section 4 afin de souligner l'indépendance de notre schéma d'extraction au regard des techniques de visualisation.

1 État de l'Art

Avant de proposer nos modifications afin d'adapter le rendu volumique direct ordonné aux maillages multirésolution, nous présentons dans un premier temps l'intégrale du rendu volumique permettant de simuler le rendu d'un milieu semi-transparent dans un espace discrétisé (section 1.1). Puis, nous dressons un état de l'art des techniques existantes permettant de la calculer : par lancer de rayons (section 1.2), par projection de cellules (section 1.3) ou par des approches hybrides (section 1.4). Nous comparons les avantages et les inconvénients de chaque type de techniques au sein de la dernière section (section 1.5).

1.1 Intégrale du rendu volumique

1.1.1 Modèle Émission-Absorption

Afin de produire un rendu volumique direct ordonné, le maillage volumique Σ est considéré comme un milieu semi-transparent composé de particules élémentaires. En appliquant des propriétés optiques physiques d'absorption, d'émission et de diffraction à ces particules, la propagation de la lumière peut être ainsi simulée [Max95]. Le modèle le plus utilisé est le *modèle émission-absorption*. Au sein de ce modèle, chaque particule de Σ absorbe une partie du rayon de lumière qui l'intersecte et transmet la lumière restante avec sa propre émission de lumière. Il est ainsi possible d'accumuler la contribution de chaque particule le long d'un rayon lumineux virtuel et de simuler l'intensité lumineuse qui parvient jusqu'à l'œil de l'observateur. Dans le cadre de l'informatique graphique, l'œil de l'observateur correspond en fait à l'écran pixélisé de l'ordinateur. Soit $\mathbf{s}(t)$ une paramétrisation d'un rayon lumineux issu d'un pixel de l'écran, l'intensité lumineuse I l'atteignant est définie par :

Définition 4.1 Intégrale du rendu volumique :

$$I = \int_0^D E(t)\, e^{-A(t)}\, \mathrm{d}t$$

$$\text{avec : } \begin{cases} E(t) = i(\mathbf{s}(t)) \\ A(t) = \displaystyle\int_0^t \alpha(\mathbf{s}(u))\, \mathrm{d}u \end{cases}$$

où D est la longueur du rayon intersectant le volume, $i(\mathbf{s})$ l'intensité lumineuse émise et $\alpha(\mathbf{s})$ l'intensité lumineuse absorbée.

Le terme $E(t)$ correspond à l'émission ponctuelle du rayonnement de chaque particule. Le terme $A(t)$ correspond à l'atténuation par absorption de l'ensemble des particules traversées entre l'émission initiale et l'œil de l'observateur. La figure 4.1 illustre ce procédé pour l'ensemble de données *Engine*.

Au sein de la visualisation scientifique (et globalement de l'informatique graphique), l'émission et l'absorption sont en fait définies via l'élaboration d'une *fonction de transfert* ϕ . La fonction d'émission $E(t)$ est en pratique un triplet $\mathbf{c}$ dans l'espace des couleurs RVB (Rouge, Vert, Bleu ou *RGB* pour *Red, Green, Blue*) [FVDF96] pour chaque particule. De la même façon, l'absorption est un coefficient d'opacité τ en chaque particule. On définit ainsi la fonction de transfert comme :

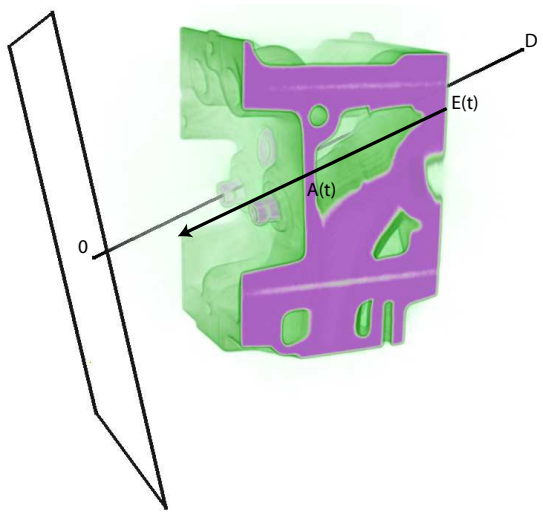


FIG. 4.1: **Intégrale du rendu volumique dans le cadre du modèle émission-absorption**. L'émission de l'intensité lumineuse $E(t)$ est partiellement absorbée en fonction de $A(t)$ jusqu'au pixel de l'écran.

Définition 4.2 Fonction de Transfert :

$$\begin{array}{rccc} \phi: & \mathbb{R} & \rightarrow & [0,1]^4 \\ & \mathbf{s} & \rightarrow & <\mathbf{c}(\mathbf{s}), \tau(\mathbf{s})> \end{array}$$

Néanmoins, l'édition d'une fonction de transfert par un utilisateur même expérimenté reste un problème chronophage et fastidieux comme nous l'avons déjà évoqué au sein du chapitre 1, section 5.3.

Une fois la fonction de transfert déterminée, l'intégrale du rendu volumique dans le cadre de la visualisation scientifique peut donc être reformulée en modifiant la définition de l'émission et de l'absorption :

Définition 4.3 Émission et Absorption pour la Visualisation Scientifique :

$$\begin{cases} E(t) = c(\mathbf{s}(t)) \\ A(t) = \displaystyle\int_0^t \tau(\mathbf{s}(u))\ \mathrm{d}u \end{cases}$$

1.1.2 Intégration Numérique

Les expressions précédentes sont définies dans le domaine continu ($\mathbb{R}$) mais n'ont pas de solution analytique. L'intégration numérique nécessite la discrétisation du rayon en un certain nombre n de segments de longueur identique $\Delta d = \frac{D}{n}$. La définition de l'intégrale du rendu volumique après discrétisation devient alors, via la transformation de l'intégrale en somme de Riemann :

Définition 4.4 Intégrale du rendu volumique discrétisée :

$$I \approx \sum_{i=0}^{n-1} E(i\Delta d)\ e^{-A(i\Delta d)}\ \Delta d$$

$$\text{avec :} \begin{cases} E(i\Delta d) = c(\mathbf{s}(i\Delta d)) \\ A(i\Delta d) \approx \displaystyle\sum_{j=0}^{i-1} \tau(\mathbf{s}(j\Delta d))\ \Delta d \end{cases}$$

On suppose généralement que le pas d'intégration Δd est suffisamment proche de 0 pour que les termes exponentiels de l'absorption soient approximés par les deux premiers termes de leur développement de Taylor. De plus, on introduit pour des raisons de clarté les termes d'émission C_i et d'opacité α_i définis par les approximations suivantes :

$$\left| \begin{array}{l} C_i \approx c(\mathbf{s}(i\Delta d))\ \Delta d \\ \alpha_i \approx \tau(\mathbf{s}(i\Delta d))\ \Delta d \end{array} \right.$$

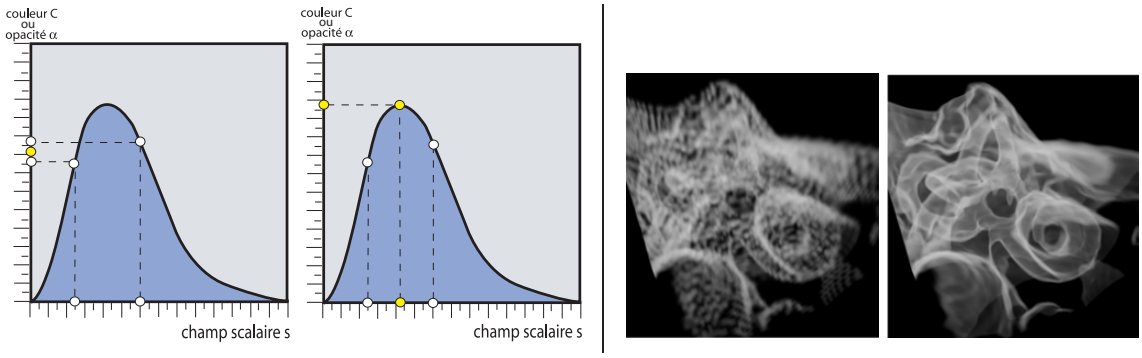


FIG. 4.2: **Pré- et Post-Classification.** Pour chaque sous-figure, la pré-classification est à gauche, la post-classification à droite. À gauche : la courbe représente la fonction de transfert ϕ. On transforme le champ scalaire en couleur et opacité. Le sommet à interpoler se situe au barycentre des deux points blancs. L'interpolation résultante est donnée en jaune. Dans le cadre de la pré-classification, on interpole les couleurs et l'opacité, alors que en post-classification, on interpole le champ scalaire en premier. On remarque que les résultats sont complètement différents. À droite : différence de rendu entre la pré-classification et la post-classification.

Cela permet d'aboutir à la formulation discrète de l'intégration du rendu volumique :

Définition 4.5 Intégrale du rendu volumique discrète :

$$I \approx \sum_{i=0}^{n-1} C_i \prod_{j=0}^{i-1} (1-\alpha_j)$$

D'un point de vue discret, l'intensité perçue par le pixel à l'écran est une somme des contributions de chacun des segments le long d'un rayon. Afin de calculer cette intensité, les contributions sont accumulées au sein d'une étape dite de *composition*. Néanmoins, il est possible de faire cette accumulation de l'avant vers l'arrière (*front-to-back*) ou de l'arrière vers l'avant (*back-to-front*). Cela donne deux modes de composition différents [Max95].

Définition 4.6 Composition de l'arrière vers l'avant :

$$C'_i = C_i + (1-\alpha_i)C'_{i+1}$$

où C'_i est l'émission accumulée entre les segments $n-1$ et i.

Définition 4.7 Composition de l'avant vers l'arrière :

$$\left| \begin{array}{l} C''_i = C''_{i-1} + (1-\alpha''_{i-1})C_i \\ \alpha''_i = \alpha''_{i-1} + (1-\alpha''_{i-1})\alpha_i \end{array} \right.$$

où C''_i et α''_i sont l'émission et l'opacité accumulées entre les segments 0 et i.

L'étape de composition n'est pas commutative. Cela a pour conséquence que l'ordre d'accumulation ne peut être changé et donc qu'un tri exact entre les segments est obligatoire afin que le rendu volumique soit exact. Cette contrainte devra donc être respectée lors de l'élaboration d'algorithmes implantant le rendu volumique direct ordonné.

1.1.3 Interpolation et Classification

Lors de l'intégration du rayon, les points d'échantillonnage définissant les segments sont indépendants des sommets du maillage Σ. La valeur du champ scalaire est donc inconnue en ces points. Des schémas d'*interpolation* adaptés à l'irrégularité du maillage permettent d'extraire des valeurs aux points d'échantillonnage. Dans le cadre des maillages tétraédriques, les coordonnées barycentriques [Möb27] du point par rapport au

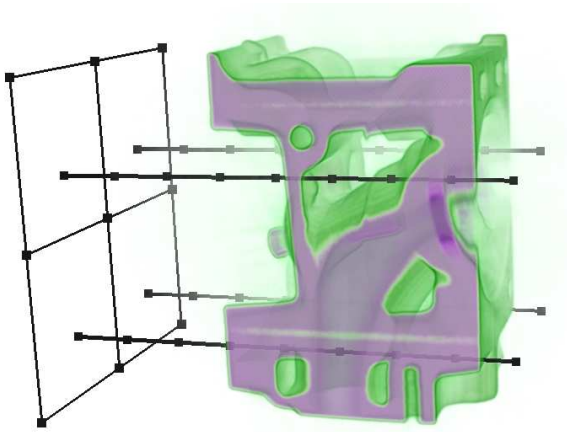


FIG. 4.3: **Lancer de Rayons**. L'image finale est composée de quatre pixels. Pour chaque pixel, un rayon est envoyé, avec un pas d'échantillonnage de huit sommets. En chacun de ces sommets, la contribution est calculée puis accumulée.

tétraèdre auquel il appartient sont généralement utilisées. Le champ scalaire est donc considéré linéaire au sein de chacune des cellules.

Comme le champ scalaire s_i est transformé par la fonction de transfert ϕ en couleur C_i et α_i, l'interpolation peut être réalisée soit sur s_i soit sur le couple $< C_i, \alpha_i >$. Dans le premier cas on parle de *post-classification*, dans le second de *pré-classification*, la *classification* correspondant à la transformation de l'espace scalaire en l'espace couleur-opacité RVBA *via* la fonction de transfert ϕ. L'application de l'une ou l'autre de ces classifications implique des résultats visuels différents comme illustrés au sein de la figure 4.2. La différence visuelle entre les deux méthodes est assez flagrante : la pré-classification introduit du bruit et des artefacts alors que la post-classification fait mieux ressortir les grandes variations du gradient du champ scalaire. Néanmoins, la pré-classification est utile lorsque les données sont segmentées. Le lecteur désirant plus de détails sur ces deux modes de classification peut se référer à [HKRS+06].

La principale limitation des fonctions de transfert est qu'elles introduisent des hautes fréquences au sein du champ scalaire lors de l'étape de rendu provoquant des artefacts. Afin de pallier à ces inconvénients, un suréchantillonnage est nécessaire mais celui-ci est coûteux. Pour l'éviter, des tables de classification pré-intégrées ont été proposées [EKE01].

Suite aux étapes d'interpolation et de classification, la couleur et l'opacité sont accumulées le long du rayon. Nous détaillons les différentes techniques [SCCB05] mises en œuvre pour le calcul de cette accumulation dans le reste de cette section.

1.2 Lancer de Rayons

Afin de réaliser l'accumulation de l'intensité des cellules pour un maillage tétraédrique, une solution est d'implanter dans l'espace image, un *lancer de rayons* (ou *ray-casting*) [Lev88]. Le calcul de l'intégrale du rendu volumique est dans ce cas-là explicite puisque l'algorithme lance pour chaque pixel de l'image finale un rayon qui est échantillonné par des segments. Le principe de l'algorithme est représenté au sein de la figure 4.3 pour une image composée de quatre pixels avec huit pas d'échantillonnage. Les rayons peuvent être parcourus de l'arrière vers l'avant ou de l'avant vers l'arrière en appliquant les formules de composition adaptées.

Des optimisations sont possibles afin d'améliorer la complexité d'un tel algorithme [Lev90]. La *terminaison précoce de rayons* (*early ray termination*) permet, lorsque le rayon est parcouru de l'avant vers l'arrière, d'arrêter l'accumulation quand le canal d'opacité est devenu (presque) opaque, (c'est-à-dire $\alpha_i'' \sim 1$). L'*accélération dans les espaces vides* (*empty space skeeping*) utilise des structures de données hiérarchiques afin d'accélérer le parcours du rayon dans les zones vides (c'est-à-dire ne comprenant pas de cellules) du maillage.

Pour les grilles non structurées, Garrity [Gar90] a proposé la première implantation de cette méthode en se basant à la fois sur une structure de données spatiale hiérarchique et les relations d'adjacence des cellules. De nombreuses améliorations ont par la suite été élaborées : [BKS97] propose de réaliser un tri des faces de bord par rapport à leur projection au sein de l'écran de façon à repérer rapidement les espaces vides ; [WKME03] réalise la première implantation sur carte graphique en stockant le maillage sous forme de textures bi- et tridimensionnelles ; [WMKE04, BPCS06] utilisent enfin la technique de *depth-peeling* afin de pouvoir traiter sur GPU des maillages tétraédriques non convexes.

Simultanément aux dernières améliorations sur la carte graphique, le parcours des rayons et leur intersection avec des primitives tétraédriques ou hexaédriques a été accéléré au sein du processeur central grâce à l'utilisation des coordonnées de Plücker [MS06] afin d'atteindre un rendu à la seconde (1 fps) pour le million de tétraèdres.

1.3 Méthodes projectives

Contrairement au lancer de rayons qui est une méthode explicite du calcul de l'intégrale du rendu volumique dans l'espace image, les approches dites *projectives* sont des méthodes implicites dans l'espace objet. Elles consistent à projeter les cellules du maillage sur l'écran afin d'en réaliser l'accumulation. Pour cela, un ordre exact des cellules est nécessaire afin d'assurer que l'intégration soit correcte lors de la projection. Nous détaillons dans un premier temps les différents algorithmes permettant d'effectuer un *ordre de visibilité* des cellules (section 1.3.1) puis les possibles méthodes de projection (section 1.3.2).

1.3.1 Tri des cellules

Ordre de Visibilité Afin de calculer l'intégrale du rendu volumique, un ordre de visibilité est nécessaire entre les différents tétraèdres pouvant être projetés au sein d'un même pixel. Cet ordre de visibilité est calculé dans l'espace objet en fonction de la direction du regard de l'observateur (ou en informatique graphique, la direction de la caméra de rendu) – cf. figure 4.4.

Pour définir cet ordre de visibilité, une relation d'ordre est nécessaire.

Définition 4.8 Ordre partiel de visibilité : On définit une relation d'ordre partiel $\leq_{\mathbf{v}}$ sur les tétraèdres telle que : $t_1 \leq_{\mathbf{v}} t_2$ si t_1 et t_2 sont adjacents (*i.e.* partagent une face) et que t_2 cache t_1 selon la direction du point de vue de l'observateur $\mathbf{v}$. On dit que t_2 est un *occludant* et que t_1 est un *occludé*.

Un ordre total de visibilité sur Σ est ainsi un ordre reposant sur la relation d'ordre $\leq_{\mathbf{v}}$. Malheureusement, comme $\leq_{\mathbf{v}}$ est un ordre partiel, il existe des maillages qui ne peuvent avoir d'ordre total selon certains points de vue $\mathbf{v}$. Un exemple classique où un ordre de visibilité est impossible à établir est donné au sein de la figure 4.5. Les trois tétraèdres t_1, t_2 et t_3 représentent un *cycle de visibilité* puisque $t_1 \leq_{\mathbf{v}} t_2$, $t_2 \leq_{\mathbf{v}} t_3$ et $t_3 \leq_{\mathbf{v}} t_1$.

Tri des tétraèdres Plusieurs algorithmes ont été proposés afin de déterminer un ordre total (lorsque celui-ci existe) pour les maillages tétraédriques. La famille des algorithmes dit *MPVO* (Meshed Polyhedra Visibility Ordering) permet de calculer l'ordre (total) de visibilité pour des maillages tétraédriques (plus généralement pour des maillages hybrides). Williams *et al.* [Wil92] propose la version initiale de cet algorithme en l'appliquant à des maillages convexes. Elle est composée de quatre étapes distinctes :

(i) *Détermination du graphe d'adjacence du maillage*. Les nœuds représentent les tétraèdres. Deux nœuds sont reliés par une arête si et seulement s'ils possèdent une face commune ;

(ii) *Détermination d'une direction par arête*. La normale du plan de la face correspondant à une arête dans le graphe d'adjacence est assignée comme direction de l'arête.

(iii) *Tri topologique du graphe*. En considérant la direction $\mathbf{v}$ du point de vue, on peut ainsi déterminer un tri topologique du graphe en appliquant un produit scalaire entre $\mathbf{v}$ et la direction des arêtes. Les cellules ne cachant aucune cellule sont nommées *source*, celles n'étant pas cachées *puits*.

(iv) *Parcours en profondeur à partir des cellules puits*. Le parcours en profondeur démarre des cellules *puits* en utilisant le tri topologique.

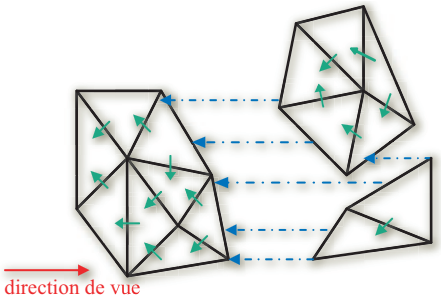


FIG. 4.4: **Ordre de Visibilité**. Il est représenté pour un maillage triangulaire sans perte de généralité. La direction de vue est donnée par la flèche rouge. L'ordre partiel $\leq_{\mathbf{v}}$ est représenté par les flèches vertes et bleues. Les vertes sont déduites par les relations d'adjacences, les bleues en pointillé sont les relations calculées pour les parties non convexes et connexes.

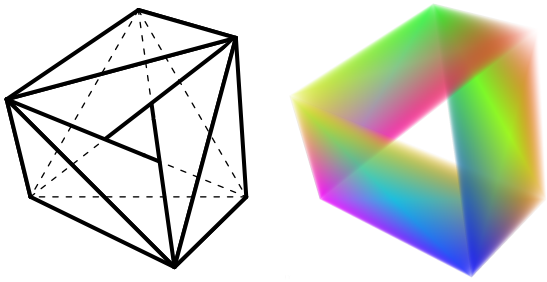


FIG. 4.5: **Cycle de Visibilité** extrait de [KE01] entre trois tétraèdres représentés en filaire puis avec un rendu volumique direct ordonné. Pour cet exemple, [KE01] transforme le maillage initial en maillage convexe (grâce à l'ajout d'un tétraèdre imaginaire) afin d'obtenir des relations de visibilité selon les faces.

Les étapes (i) et (ii) peuvent être précalculées afin d'optimiser les performances. De plus, lors du parcours en profondeur, les cycles de visibilité peuvent être détectés. Néanmoins, cette version est limitée aux maillages convexes.

Plusieurs améliorations ont ainsi été apportées pour gérer à la fois la non-convexité des maillages et diminuer les complexités mémorielle et temporelle de l'approche. Pour cela une étape supplémentaire est insérée lors du tri topologique détectant les faces de bord du maillage et les triant d'une manière plus ou moins efficace : par lancer de rayons [SMW98, CMSW04] ou l'utilisation de BSP-Tree [CKM+99]. Ce tri permet ainsi de gérer les espaces vides entre les différentes parties du maillage qui pourraient se cacher mutuellement.

Enfin, les cycles de visibilité ont été gérés par [KE01] – cf. figure 4.5.

La complexité temporelle de tels algorithmes ne permet pas de réaliser un rendu volumique direct temps-réel (au sens donné au sein du chapitre 1, section 3.2). Pour de nombreux rendus, un ordre total inexact est souvent préféré basé sur le tri des barycentres [Luc92] ou la distance *puissance*[1] [CDF98]. Ceux-ci peuvent ainsi être réalisés en $O(card\ T\ log_2(card\ T))$ grâce à un tri optimisé comme le tri par base (*radix sort*) où T est le nombre de tétraèdres de Σ. Néanmoins, l'insertion d'erreurs au sein de l'ordre de visibilité peut introduire un certain nombre d'artefacts visuels qui peuvent perturber (plus ou moins) l'utilisateur lors de son exploration.

Accélération Graphique Pour accélérer ces tris, la carte graphique a été récemment utilisée pour réaliser une partie des calculs. L'idée générale de ces améliorations architecturales est de réaliser une partie voire la totalité du tri sur la carte graphique. Carpenter fut le premier à proposer une telle architecture afin de pouvoir réaliser un tel tri grâce à son *A-Buffer* [Car84], décrit comme une extension du tampon de profondeur (*z-buffer*). Le A-Buffer permet ainsi de trier les primitives et de réaliser ensuite l'accumulation de celles-ci pour produire un rendu volumique direct ordonné correct. Il fut néanmoins implanté sur CPU. Avec la création des cartes graphiques programmables, une première tentative de portage du A-Buffer de Carpenter sur GPU : le *R-Buffer*, a été proposée [Wit01]. À cause des limitations architecturales, le R-Buffer ne pouvait pas stocker sur la carte graphique l'ensemble des fragments pour un seul et même pixel.

Pour pallier à cet inconvénient, deux autres méthodes ont été développées. La première est la technique des cartes de profondeur ou *depth-peeling* proposée par Everitt [Eve01] et améliorée plusieurs fois en fonction de l'évolution des cartes graphiques (dont la dernière [BM08]). Le *depth-peeling* utilise le rendu hors-écran afin de calculer les couches de profondeur du volume à partir du point de vue courant. Un certain nombre de passes est ainsi réalisé, correspondant à la profondeur maximale de l'ordre de visibilité. Cette méthode bien que réalisée dans l'espace image garantit que l'ordre établi est un ordre total s'il n'y a pas de cycle de visibilité [CMSW04]. Krishnan *et al.* propose une méthode équivalente pour classifier les triangles des surfaces de bord afin de gérer

[1] La distance *puissance* garantit un ordre de visibilité exact pour les maillages de Delaunay

plus rapidement le tri de ceux-ci lorsque le maillage n'est pas connexe ou convexe [KSW01]. Notons que les auteurs ne parlent pas de *depth-peeling* au sein de leur implantation bien que leur méthode est similaire.

La seconde méthode est le système nommé *HAVS*[2](*Hardware-Assisted Visibility Sorting*) developpé par Callahan *et al.* [CICS05]. Il repose sur l'implantation d'un *k-buffer* au sein de la carte graphique permettant de trier en une seule passe jusqu'à k fragments par pixel, k étant dépendant de l'architecture de la carte graphique et est fixé actuellement à $k = 6$. Afin d'assurer que k comparaisons seront au plus nécessaires par pixels, les faces du maillage tétraédrique sont préalablement triées partiellement par rapport au point de vue grâce à un tri par base en mémoire centrale. Elles sont ensuite transmises (en tant que listes presque triées par pixel) à la carte graphique qui peut appliquer le k-buffer et calculer un ordre de visibilité. Une telle solution permet d'utiliser pleinement la puissance de calculs des cartes graphiques et ainsi assure une interactivité jusqu'alors inégalée. Néanmoins, l'implantation ne peut garantir pour tous les pixels que les faces seront exactement triées. En effet, le tri partiel est réalisé dans l'espace objet (non dans l'espace image) et ne peut garantir que k échanges de fragments seront seulement nécessaires pour obtenir un ordre total dans le pixel courant. De telles erreurs peuvent ainsi créer des artefacts pouvant perturber l'utilisateur lors de son exploration.

Citons enfin un tri implanté sur carte graphique pour des scènes géométriques avec transparence qui pourrait être aussi utilisé pour calculer l'ordre de visibilité des tétraèdres [GHLM05].

1.3.2 Projection

Les algorithmes de projection pour les maillages tétraédriques sont inspirés de la méthode de *splatting* [Wes90] développée originalement pour les grilles régulières. Chaque voxel d'une grille régulière est remplacée par une empreinte issue de sa projection dans le plan de l'image. Cette empreinte est souvent un noyau gaussien. Par la suite, nous ne détaillons uniquement que les méthodes de projection associées au rendu de maillages tétraédriques.

Projected Tetrahedra L'une des techniques les plus utilisées est la méthode de projection développée par Shirley et Tuchman nommée *Projected Tetrahedra* [ST90]. Elle consiste à décomposer le tétraèdre en un certain nombre de faces en fonction du point de vue. La figure 4.6 illustre les quatre cas différents possibles. Un tétraèdre peut ainsi se décomposer en un certain nombre de triangles de un à quatre en fonction de son orientation par rapport au point de vue. Le calcul exact de l'intégrale du rendu volumique est approximé en utilisant l'épaisseur du tétraèdre au sommet (possiblement factice) où celle-ci n'est pas nulle.

L'implantation de cette projection au sein de la carte graphique a été proposée *via* l'utilisation d'un processeur de sommets (*GATOR*) [WMFC02] puis améliorée avec deux passes utilisant le processeur de fragments (*PTINT*) [MMFE06]. Celle-ci est réalisable grâce à l'utilisation d'un graphe *de base* représentant de manière générique la projection d'un tétraèdre dans l'espace image. Ce graphe est composé de cinq sommets et de quatre triangles. La dégénérescence de ce graphe en répétant certains sommets permet ainsi de traiter l'ensemble des possibilités de projection au sein de la carte graphique. Nous proposons par la suite notre propre amélioration de cette méthode très utilisée au sein des logiciels de visualisation de maillages tétraédriques reposant sur la nouvelle génération de cartes graphiques et le processeur de primitives programmable.

Rendu par points Une autre alternative à la projection des tétraèdres est de remplacer ceux-ci par des points. Ainsi, [ML04] propose un découpage du maillage entre la surface de bord et l'intérieur afin de créer des structures hiérarchiques de sphères englobantes. Ces deux zones sont rendues avec des disques opaques dont la taille est adaptée en fonction du taux de rafraîchissement et de la précision voulue. Un disque peut correspondre à un tétraèdre, un triangle ou un sommet. Des coupes sont possibles afin de voir l'intérieur du maillage.

[ACS+07] remplace les triangles utilisés au sein de HAVS par des points. Il propose une construction à la volée de niveaux de détails permettant d'ajuster dynamiquement le rendu. Les points sont ensuite transmis à la carte graphique pour que leur rayon et leur forme (sphère, ellipse) soient adaptés afin de réaliser la composition finale.

[2]Le code source de *HAVS* est disponible à l'adresse suivante : http ://havs.sourceforge.net/ et a été implanté en tant que greffon dans *Paraview*.

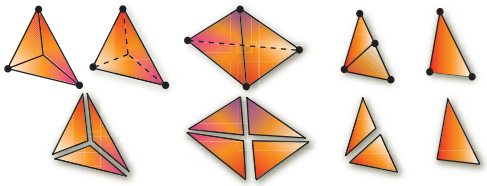


FIG. 4.6: **Projected Tetrahedra**. En haut, les tétraèdres dans l'espace objet. En bas, la décomposition correspondante en triangles. Un tétraèdre est décomposable selon le point de vue en quatre cas distincts : les deux premiers donne trois triangles, le suivant quatre. Les deux derniers sont des cas dégénérés donnant deux et un triangles respectivement.

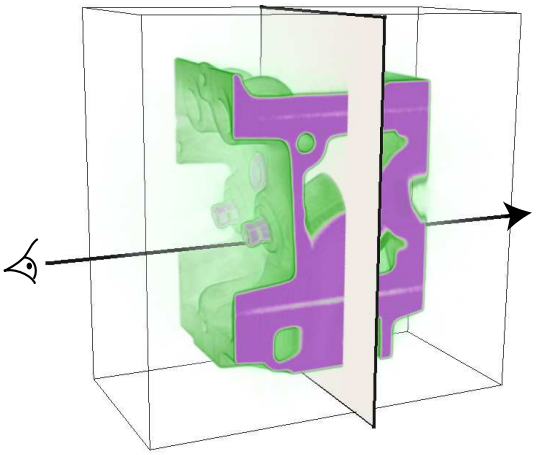


FIG. 4.7: **Paradigme de Balayage**. Un plan de balayage parcourt l'espace, ici le long de la direction de vue. Les cellules intersectant le plan sont ordonnées puis composées afin de contribuer au rendu final.

1.4 Méthodes Hybrides

Une dernière famille de méthodes repose sur un mélange de projection et de lancer de rayons. Elle s'inspirent à l'origine de l'algorithme de rendu *scanline* qui traite l'affichage, ligne horizontale par ligne horizontale, sur l'écran d'un ordinateur et qui a été adapté pour calculer le rendu volumique direct ordonné pour des maillages irréguliers [WVGTG96]. Ces algorithmes se basent ainsi sur le paradigme du balayage (ou *sweeping paradigm*).

Paradigme du Balayage Le but est de remplacer la ligne de balayage par un plan de balayage dont l'orientation varie selon les implantations. L'espace objet est ainsi parcouru par ce plan selon l'axe choisi et les intersections entre le plan et les cellules du maillage sont calculées par projection – cf figure 4.7. Cela permet ainsi de réaliser un lancer de rayons bidimensionnel tout en triant la dernière coordonnée selon le balayage du plan [Gie92]. Cet algorithme a connu de nombreuses améliorations afin d'utiliser la carte graphique [RYL+96], gérer les maillages non convexes et non connexes [SM97] et enfin en diminuer la complexité mémorielle et temporelle [FMS00].

1.5 Comparaisons et Discussions

Différents rendus de l'ensemble de données *SPX* sont présentés au sein de la figure 4.8 et les temps associés sont renseignés dans le tableau 4.1. Quatre méthodes ont été utilisées, celles implantées au sein du logiciel de visualisation *Paraview*[3] : *Projected Tetrahedra* (*PT*) [ST90] et *HAVS* [CICS05] pour les méthodes de projection, le lancer de rayons de Bunyk (*RC*) [BKS97] sur CPU, et *ZSWEEP* [FMS00] pour les méthodes hybrides. Nous discutons les temps de rendu ainsi que la qualité de ceux-ci.

Comparaisons Concernant la complexité temporelle des rendus, les méthodes de projection restent les plus rapides en toutes circonstances, l'accélération graphique permettant de conserver le temps-réel pour des maillages dont la taille est supérieure à un million de cellules. Le lancer de rayons au sein du processeur central reste interactif (comme défini au sein du chapitre 1, section 3.2) et le temps-réel est atteint en utilisant une version transposée sur carte graphique (1-2 affichage(s) à la seconde pour le million de tétraèdres) comme celle

[3] disponible à l'adresse : http ://www.paraview.org.

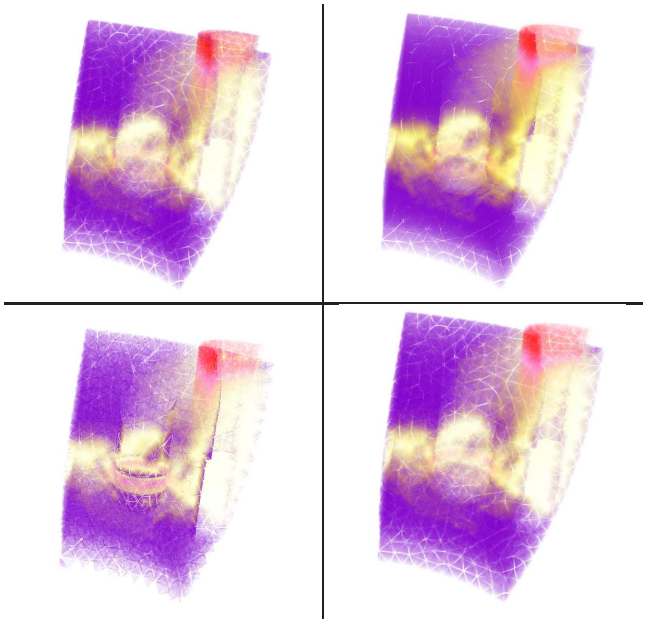


FIG. 4.8: **Comparaisons des techniques de rendu volumique direct ordonné**. En haut à gauche : *PT* [ST90]. En haut à droite : *HAVS* [CICS05]. En bas à gauche : *ZSWEEP* [FMS00]. En bas à droite : *RC* [BKS97].

Données	Rendu Volumique Direct			
	PT	*HAVS*	*ZSWEEP*	*RC*
SPX	0,031	0,059	3,783	3,044
Comb	0,422	0,125	29,36	6,211
Torso	2,104	0,344	107,3	10,75

TAB. 4.1: **Temps d'affichage des techniques de rendu volumique direct ordonné**. Les méthodes comparées sont *PT* [ST90], *HAVS* [CICS05], *ZSWEEP* [FMS00] et *RC* [BKS97]. Les temps en seconde sont moyennés suite à plusieurs points de vue, l'image générée est composée de 800×800 pixels.

de Bernardon *et al.* [BPCS06][4]. L'algorithme hybride est très lent.

Comparons maintenant la qualité des rendus entre les différents algorithmes. Nous signalons que le choix de la fonction de transfert est d'une importance capitale pour la qualité des images. Ainsi, une fonction de transfert adaptée pour une méthode pourrait se révéler génératrice d'artefacts pour une autre – cela est dû à l'approximation discrète qui va selon les méthodes modifier spatialement l'opacité selon les pixels ou les cellules. Il est donc important de savoir adapter une telle définition en fonction des méthodes de rendu utilisées.

Néanmoins, dans tous les cas, la méthode de projection reposant sur *Projected Tetrahedra* et le lancer de rayons assure des rendus de meilleures qualités que les deux autres. Notons que le lancer de rayons est un peu plus diffus le long des arêtes vives du maillage.

Dans le cadre de notre comparaison, nous avons fixé une fonction de transfert et un ensemble de données. Notons ainsi, que *HAVS* génère un ordre de visibilité par pixels qui n'est pas exact partout ; et que *ZSWEEP* dissocie, avec le choix de cette fonction de transfert, trop fortement les différentes cellules créant des artefacts nuisibles à la qualité du rendu final. Dans tous les cas, les tétraèdres sont visibles, le maillage n'étant pas composé d'assez de cellules pour permettre une accumulation diffuse pour tous les pixels.

Pour conclure, les méthodes de projection semblent donc être les meilleures candidates alliant faible complexité temporelle et qualité d'images pour réaliser un rendu volumique direct ordonné.

Discussions Les algorithmes de rendu volumique direct travaillent sur des maillages tétraédriques statiques, c'est-à-dire qu'ils ne sont pas implantés pour gérer de manière efficace des maillages dont la connectivité et la géométrie sont modifiées à chaque rendu, en d'autres termes des maillages multirésolution. En effet, la multirésolution est souvent spécifique à de tels rendus – au lieu d'adapter la technique de rendu à des algorithmes multirésolution existants. Ainsi, Farias *et al.* [FMSW00] propose une approche multirésolution à la fois dans l'espace image (en regroupant les rayons par groupe de $n \times n$ pixels) et dans l'espace objet (avec une hiérarchie statique de maillages simplifiés). Callahan *et al.* [CCS05] proposent différentes méthodes d'échantillonnage des faces envoyées à la carte graphique afin de réaliser des niveaux de détails tout en minimisant l'erreur visuelle. Cette idée est reprise pour le rendu par points [ACS+07].

[4] *HRC* est disponible à l'adresse suivante : http ://www.cs.utah.edu/ csilva/software/gpu_volume_ray_casting.zip.

Ainsi, à notre connaissance, aucun algorithme de rendu volumique n'a tenté de s'adapter aux schémas de multirésolution existants. Dans la suite de ce chapitre, nous proposons des modifications d'algorithmes existants afin que ceux-ci prennent en comptent efficacement les modifications de connectivité dynamiques liées à notre approche birésolution.

Les méthodes de lancer de rayons n'étant efficaces que si elles transmettent à la carte graphique le maillage sous des formats demandant quelques précalculs et les méthodes hybrides étant assez lentes, nous avons opté pour les méthodes reposant sur la projection plus propices à s'adapter aux maillages multirésolution et assurant de plus un rendu d'une certaine qualité.

2 Tri des primitives et cohérence temporelle

2.1 Objectifs

Dans le cadre de l'implantation de notre méthode d'extraction au sein de la mémoire centrale, notre premier choix fut d'adapter et d'améliorer l'algorithme de tri de Cook *et al.* [CMSW04] nommée *SXMPVO* (pour *Scanning eXact Meshed Polyhedra Visibility Ordering*) permettant de réaliser un ordre de visibilité pour les maillages tétraédriques non connexes et non convexes. Nous avons inséré au sein de *SXMPVO* la cohérence temporelle introduite lors de l'extraction du maillage birésolution (cf. chapitre 3, section 2.2), afin que, lorsque la région d'intérêt est déplacée mais pas le point de vue, la complexité de l'algorithme de tri soit réduite. Nous développons cette amélioration dans la section 2.2.

Une fois l'ordre de visibilité déterminé, nous projetons les tétraèdres en utilisant la méthode *Projected Tetrahedra* proposée par Shirley et Tuchman [ST90]. Avec l'émergence des nouvelles cartes graphiques et la possibilité de programmer le processeur de primitives, cette méthode de projection se révèle parfaitement adaptée pour être transposée au sein de la carte graphique. Nous proposons notre propre transposition de cette méthode au sein de la section 2.3.

Nous discutons enfin les résultats en section 2.4.

2.2 SXMPVO Cohérent

Nous proposons une amélioration de l'algorithme *SXMPVO* afin que celui-ci inclut la cohérence temporelle dans le cadre d'un rendu volumique appliqué à un maillage birésolution. Dans un premier temps, on rappelle l'algorithme original (section 2.2.1). Puis, nous détaillons les modifications apportées afin d'introduire la cohérence temporelle (section 2.2.2). Enfin, nous proposons d'utiliser la carte graphique afin d'accélérer l'algorithme (section 2.2.3).

2.2.1 Algorithme original

L'algorithme *SXMPVO* [CMSW04] garantit un ordre total de visibilité pour les maillages non convexes et non connexes sans cycle. Il est composé de quatre étapes :

Étape I : Elle détermine un ordre partiel de visibilité en utilisant les relations d'adjacence dans les zones connexes. Pour cela, le signe du produit scalaire entre la direction de vue et les normales de la face commune entre deux tétraèdres permet de trouver l'occludant et l'occludé. Chaque tétraèdre peut ainsi se situer dans l'ordre de visibilité par rapport à ses voisins ;

Étape II : Elle trie les faces de bord selon leur profondeur afin de pouvoir déterminer les relations d'occlusions entre les parties non connexes et non convexes dans l'étape suivante ;

Étape III : Elle détermine l'ordre de visibilité des faces de bord dans l'espace objet *via* un lancer de rayons. Les rayons sont stockés dans un A-Buffer [Car84]. Le A-Buffer mémorise pour chaque pixel de l'image une liste ordonnée en fonction de la profondeur des faces que le rayon associé intersecte. En enlevant pour chaque pixel la première face intersectée, on obtient alors des couples ordonnés de faces, la première cachant la seconde. Cela permet ainsi de construire les relations de visibilité manquantes.

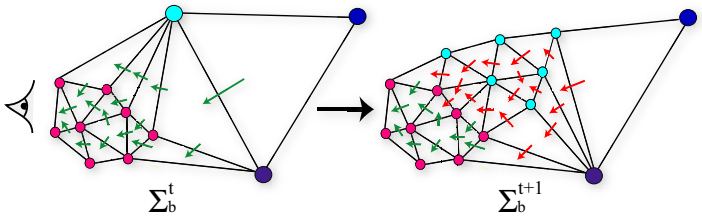


FIG. 4.9: **Mise-à-jour des relations de visibilité au sein du maillage birésolution.** L'utilisateur regarde vers la droite et ne bouge pas. À gauche : ordre de visibilité courant représenté par les flèches vertes. À droite : la boule d'intérêt est déplacée (non représentée). Le cluster associé au sommet grossier cyan devient actif. De nouveaux triangles actifs sont ajoutés. Une partie des relations de visibilité est inchangée (flèches vertes), une autre partie doit être mise à jour (flèches rouges).

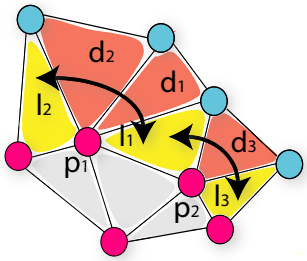


FIG. 4.10: **Détermination des relations d'adjacence entre tétraèdres de lien**. On cherche les relations d'adjacence de l_1. Pour cela, on utilise tous les triangles de lien, conservés en jaune et dégénérés en rouge. La recherche consiste à pivoter autour des sommets p_1 et p_2. Autour de p_1, les relations l_1, d_1, d_2, l_2 permettent de trouver que l_1 est adjacent à l_2. De même autour de p_2, l_1 est adjacent à l_3.

Étape IV : Elle réalise un parcours en profondeur de cet ordre de visibilité total à partir des faces sources (comme définies en section 1.3.1, étape *(iii)*) afin d'afficher en utilisant le GPU l'ensemble des primitives de l'arrière vers l'avant.

Nous détaillons dans la suite deux améliorations : l'insertion de la cohérence temporelle au sein du calcul de l'ordre de visibilité (étapes I et IV) et une accélération graphique pour remplacer le A-Buffer (étapes II-III).

2.2.2 Algorithme Cohérent

Tout d'abord, nous proposons d'utiliser la mise-à-jour de la région d'intérêt (décrite au chapitre 3, section 2.2) pour accélérer une partie de l'algorithme *SXMPVO* dans la situation où l'utilisateur a fixé son point de vue. En effet, le rendu volumique étant dépendant du point de vue, l'ordre de visibilité doit être recalculé pour toutes les primitives du maillage birésolution si celui-ci est modifié. Si ce n'est pas le cas, alors des améliorations sont envisageables grâce à la cohérence temporelle. Dans le reste de cette section, on supposera donc que l'utilisateur a fixé son point de vue et déplace uniquement la zone d'intérêt, scénario réaliste lors de l'exploration des données. La cohérence temporelle est alors introduite au sein des étapes I et IV de l'algorithme original (section 2.2.1).

Modification Étape I L'étape I de *SXMPVO* détermine un ordre partiel en se basant sur les relations d'adjacence. Lors d'une exploration où le point de vue n'est pas modifié, cet ordre de visibilité est inchangé si le maillage est statique. Dans le cadre d'un maillage birésolution où une boule d'intérêt est déplacée par l'utilisateur, deux types de cellules sont à considérer pour mettre à jour l'ordre de visibilité : les tétraèdres fins actifs et les tétraèdres de lien nouvellement ajoutés.

Pour les tétraèdres fins actifs nouvellement ajoutés, cette insertion est faite lors de l'étape 3 (ajout de nouveaux tétraèdres) de la mise-à-jour de la zone d'intérêt – comme spécifié au sein du chapitre 3 précédent, section 2.2. Les nouvelles cellules sont triées avec leurs voisins comme illustré au sein de la figure 4.9 au sein d'un maillage tétraédrique sans perte de généralité. De plus, lorsque les tétraèdres sont enlevés de la zone d'intérêt (étapes 1-2), leurs relations de visibilité sont effacées pour ne pas être envoyées à la carte graphique lors du parcours en profondeur.

Afin d'afficher le maillage birésolution avec un rendu volumique direct, un stockage en mémoire de l'ensemble des cellules de lien est réalisé. En effet, les relations d'adjacence entre les cellules de lien sont nécessaires pour assurer un ordre partiel correct. Elles sont mises à jour à chaque fois que la zone d'intérêt est déplacée, en plus de la détermination de l'ordre de visibilité. On peut décomposer leur traitement en trois étapes :

1. **Trouver les tétraèdres de lien**. Les tétraèdres de lien sont sauvegardés dans une table de hachage. Pour les trouver, l'ensemble des tétraèdres fins actifs est parcouru. Tous les tétraèdres de lien détectés sont

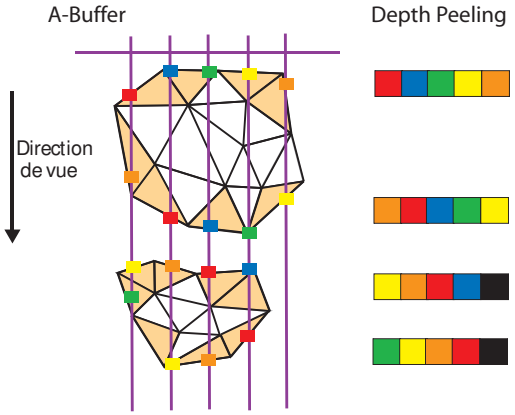


FIG. 4.11: **Correspondance entre *A-Buffer* et *Depth-Peeling*.** Les tétraèdres de bord sont orangés. Chaque intersection au niveau du *A-Buffer* est représentée par un carré de couleurs. Les textures extraites pour le *depth-peeling* sont données en correspondance.

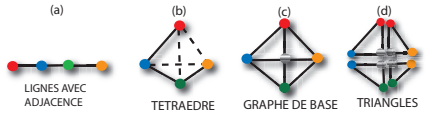


FIG. 4.12: **Implantation de *Projected Tetrahedra* (PT) au sein d'un processeur de primitives**. Les tétraèdres (b) sont représentés par des GL_LINES_ADJACENCY_EXT (a). Au sein du processeur de primitives, ils sont projetés sur le graphe de base (c) puis les triangles correspondants (d) sont envoyés au processeur de fragments.

stockés, dégénérés ou non. Les relations d'adjacence avec les tétraèdres fins et grossiers sont calculées.

2. **Mettre à jour les relations d'adjacence entre cellules de lien**. Pour cela, les relations d'adjacence entre les tétraèdres fins correspondant sont utilisés. En effet, chaque cellule de lien est en fait une cellule fine étirée ou dégénérée. Ces cellules fines sont donc visitées jusqu'à trouver une cellule fine dont la cellule de lien correspondante n'est pas dégénérée. On obtient alors une relation entre deux tétraèdres de lien (cf. figure 4.10). À la fin de cette étape, les cellules de lien dégénérées sont enlevées.
3. **Calculer l'ordre de visibilité**. Celui-ci est déterminé en utilisant les relations d'adjacence précédemment calculés de manière identique aux autres cellules.

À la fin de l'étape I, un ordre partiel est ainsi obtenu pour le maillage birésolution.

Cette mise en place de la cohérence temporelle permet de réduire la complexité de l'étape I au nombre de tétraèdres ajoutés et enlevés lors de la mise à jour – comme vu au chapitre 3, section 2.4.1.

Modification Étape IV L'étape IV est modifiée pour afficher l'ensemble des tétraèdres du maillage birésolution. Durant le parcours en profondeur, les primitives sont marquées pour éviter des rendus multiples ou détecter des cycles de visibilité. Le test de visite a été modifié afin de connaître durant quel rendu le tétraèdre a été marqué. Si la marque correspond au rendu actuel, le parcours est arrêté, sinon il correspond au rendu précédent et le tétraèdre est parcouru. Cette modification assure que tous les tétraèdres et en particulier les anciens, sont envoyés à la carte graphique.

2.2.3 Accélération Graphique

Pour accélérer *SXMPVO*, une implantation sur carte graphique de la technique de *depth-peeling* [Eve01] est réalisée pour remplacer la structure en mémoire principale du A-Buffer. Cette idée a été citée au sein des futurs travaux sans jamais être implantée pour *SXMPVO*.

En effet, le A-Buffer consiste à émuler sur le CPU un tramage des faces de bord afin de calculer un tri selon leur profondeur par rapport au point de vue. Le *depth-peeling* permet de réaliser ce tri tout en utilisant les unités de tramage disponibles au sein de la carte graphique. La figure 4.11 illustre la correspondance entre les deux méthodes. Chaque intersection avec une face de bord est représentée par un carré coloré. Les textures obtenues par *depth-peeling* sont indiquées en correspondance.

Le principe du *depth-peeling* est de réaliser des rendus successifs de la scène. Chaque rendu enlève les faces qui étaient visibles au rendu précédent. Si l'on stocke dans des textures chacun de ces rendus successifs,

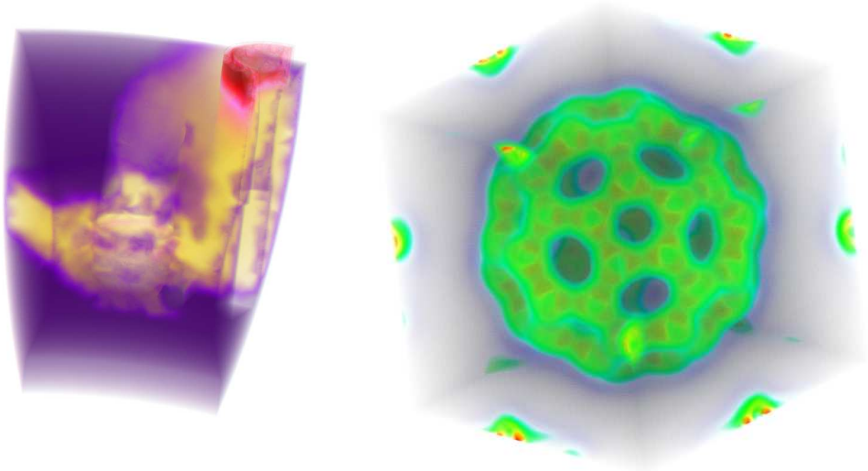


FIG. 4.13: **Exemples de rendu volumique direct avec SXMPVO accéléré et PT au sein d'un processeur de primitives**. Ces rendus sont effectués respectivement sur l'ensemble *SPX* et *BuckyBall.*

on calcule alors l'ordre suivant : les faces les plus proches, puis les deuxièmes, ..., jusqu'aux plus profondes. On obtient ainsi un A-Buffer sur carte graphique, les textures faisant office de tampons d'accumulation.

En envoyant uniquement à la carte graphique les faces de bord déduites lors de l'étape I, on peut ainsi obtenir au sein de textures l'ensemble des relations de visibilité les concernant dans l'espace image. Ces textures sont ensuite rapatriées au sein du processeur central – exceptée la première qui n'intervient pas dans l'ordre de visibilité ; et assemblées par couple pour retrouver les relations d'occlusions.

Le *depth-peeling* est implanté de manière efficace, chaque passe déterminant une couche de profondeur (*layer*) grâce à un rendu hors écran au sein du tampon de profondeur (*z-buffer*). Ce même tampon étant utilisé en entrée du rendu suivant afin de pouvoir enlever les fragments dont la profondeur serait supérieure à celle stockée dans le tampon[5]. Des tampons de textures (*pixel buffers*) sont utilisés pour accélérer le rapatriement des couches de profondeur au sein du processeur central. De plus, l'espace image est découpé en *tiles* (4×4) afin de détecter plus rapidement les parties de l'écran qui sont vides grâce à des demandes d'occlusions (*occlusion queries*) et rapatrier des textures plus petites.

Cette approche permet ainsi d'éliminer la phase II triant les faces de bord et accélère la phase III qui était le goulot d'étranglement de l'algorithme *SXMPVO* original.

2.3 *Projected Tetrahedra* au sein d'un processeur de primitives

Afin de réaliser le rendu par projection de tétraèdres, il est nécessaire de connaître la projection de ceux-ci dans l'espace image. Pour cela, *Projected Tetrahedra* projette chaque tétraèdre sur un graphe de base qui détermine le nombre de triangles résultant contribuant au rendu final. Leur nombre peut varier de un à quatre en fonction des configurations. Lorsque quatre triangles sont nécessaires, un nouveau sommet doit être introduit, déterminé par une double interpolation linéaire.

L'algorithme *GATOR* [WMFC02] est la première méthode de projection de tétraèdres implantée sur carte graphique. Elle est confrontée au problème que la connectivité était inaccessible au sein de la carte graphique, le processeur de primitives étant alors fixe. Ainsi, pour créer des triangles, ceux-ci se devaient d'être tous envoyés à la carte graphique à l'initialisation, quitte à être dégénérés par la suite lors du rendu final. De plus, chaque tétraèdre se doit d'être connu afin d'être projeté. Afin de pallier à cet inconvénient, *GATOR* propose d'envoyer pour chaque sommet appartenant aux triangles projetés, l'ensemble des informations concernant le tétraèdre courant.

La projection est faite au sein du processeur de sommets. Chaque tétraèdre est envoyé cinq fois (car cinq sommets représentent les quatre triangles possibles *via* l'utilisation de *triangle strips*) pour garantir une trans-

[5] si l'on suppose que le pixel le plus proche à une profondeur de 1 et le plus éloigné une profondeur de 0.

Image	SXMPVO Original				SXMPVO Amélioré			
512 × 512	I+ II	III	IV	R	I	III	IV	R
0,4%	195	17	136	2,8	187	12	56	3,7
3,7%	241	46	136	2,4	187	14	56	3,7
39,4%	433	269	138	1,3	180	20	57	3,7
99,3%	280	792	136	0,9	185	26	56	3,6
680×840	I+ II	III	IV	R	I	III	IV	R
0,3%	195	24	139	2,75	188	31	54	3,5
2,3%	257	87	138	2,1	185	33	55	3,5
31,1%	381	476	138	1	185	42	55	3,4
99,9%	242	1629	139	0,5	187	66	56	3,1
945×1210	I+ II	III	IV	R	I	III	IV	R
2,1%	296	122	139	1,9	187	64	55	3,2
29,3%	338	1173	137	0,7	189	78	56	3,0
99,9%	204	2962	134	0,3	187	126	55	2,6

TAB. 4.2: **Comparaisons de *SXMPVO* original et architecturalement accéléré**. 121 259 tétraèdres sont affichés. Les temps sont exprimés en millisecondes pour les étapes (I,II,III,IV) de *SXMPVO*. Le rendu global (R) est en images par seconde. Le pourcentage de l'image occupée par la projection de la zone d'intérêt est indiqué.

Données	*PTINT*	*GPTS*
Blunt	11,3 fps	14,3 fps
Comb	9,32 fps	16,6 fps
Post	4,49 fps	6,1 fps
Fuel	1,49 fps	3,22 fps

TAB. 4.3: **Comparaisons temporelles entre *PTINT* et notre implantation**. Les taux de rafraîchissement de *PTINT* sont extraits de [MMFE06]. Notre méthode (GPTS) est 1,64 fois plus rapide en moyenne.

mission sans perte au processeur de fragments. Cette approche impose ainsi l'envoi d'une grande redondance d'information à la carte graphique et réalise des calculs tout aussi redondants en son sein.

Pour améliorer cette approche, nous transformons le *vertex shader* en *geometry shader* ce qui permet d'obtenir l'information de connectivité au sein du processeur de primitives programmable et de modifier directement la géométrie de tétraèdres en triangles – cf. figure 4.12. Cela permet ainsi certaines améliorations :

- Un tétraèdre est envoyé une seule fois au lieu de cinq fois sous la forme d'une ligne avec adjacence de quatre points repérés par leur position et leur couleur. La taille des données envoyées à la carte graphique est donc grandement diminuée. Elle est égale à la taille d'un maillage représenté par une soupe de cellules ;
- Chaque tétraèdre est projeté une seule fois au sein du processeur de primitives *via* quelques modifications de l'implantation originale dans un souci d'optimisation du *shader* ;
- De plus, aucune primitive dégénérée n'est envoyée au processeur de fragments. Cette solution évite les deux passes que proposent Marroquim avec son implantation *PTINT* [MMFE06].

2.4 Résultats et Discussion

Nous évaluons dans un premier temps les accélérations graphiques (sections 2.2.3 et 2.3) puis l'insertion de la cohérence temporelle au sein de *SXMPVO* (section 2.2.2).

Accélération Graphique Le tableau 4.2 illustre l'amélioration en temps obtenue grâce à l'utilisation du *depth-peeling* (III) et du processeur de primitives *GPTS* (*Geometry Projected Tetrahedra Shader*) (IV) par rapport à une implantation originale avec le A-Buffer (III) et GATOR (IV). Aucune cohérence temporelle n'est utilisée, différentes résolutions d'images $s_x \times s_y$ sont proposées : 512×512, 680×840 et 945×1210. Les rendus obtenus sont présentés au sein de la figure 4.13.

Alors que le nombre de pixels augmente, notre nouvelle phase III devient de plus en plus rapide par rapport à l'ancienne. On peut d'ailleurs estimer un gain proportionnel $\gamma \propto \frac{s_x \times s_y}{2\,000}$ par rapport à l'implantation originale. De plus, zoomer sur la zone d'intérêt ralentit peu cette phase malgré un nombre de pixels plus important à traiter. On remarque tout de même que lorsque le nombre de pixels contribuant à l'écran est faible (en deçà des 1% de contribution à l'image finale), l'implantation sur carte graphique peut être plus chronophage que son implantation au sein du processeur central. Cela s'explique par le fait que le transfert et le rapatriement

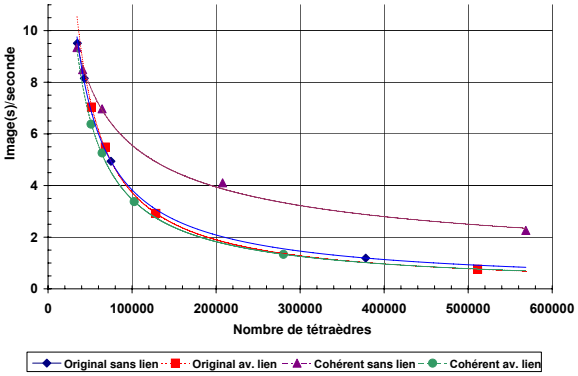


FIG. 4.14: **Comparaisons entre SXMPVO avec et sans cohérence**. On utilise dans tous les cas, l'algorithme architecturalement accéléré grâce aux dernières fonctionnalités des cartes graphiques. Les courbes représentent la distinction entre l'algorithme avec ou sans cohérence et la création d'un maillage à deux composantes connexes (grossier - zone d'intérêt, sans lien) ou unique (avec lien). La taille de la fenêtre est 680×840.

des données coûtent plus cher que le calcul de la visibilité au sein du A-Buffer dans les cas où il y a peu d'informations à traiter.

Le *geometry shader* repose sur la table de projection de *GATOR*. La table de Marroquim a aussi été testée mais les temps de rendu sont plus lents (63 au lieu de 55 ms) sans différences visuelles remarquables. L'étape IV présente ainsi un gain de $2,5$ par rapport à l'implantation original de *GATOR* [WMFC02]. Nous avons aussi comparé notre implantation sur processeur de primitives avec *PTINT* [MMFE06] utilisant deux passes au sein du processeur de fragments. Nous avons effectué un certain nombre de rendus au sein d'une fenêtre 512×512 et obtenu les taux de rafraîchissement fournis au sein du tableau 4.3. En moyenne, notre solution *GPTS* est $1,64$ fois plus rapide.

Pour conclure avec l'utilisation de la carte graphique pour les étapes III et IV et la suppression de l'étape II un gain de $3,2$ est obtenu en moyenne sur l'ensemble du pipeline de rendu.

Cohérence Temporelle Nous nous focalisons maintenant sur l'utilisation de la cohérence temporelle. La figure 4.14 montre le nombre d'images par seconde pour le rendu volumique direct ordonné en utilisant le *depth-peeling* et le *geometry shader* dans une fenêtre 680×840.

Dans un souci de clarté, nous distinguons les algorithmes original et cohérent. Nous séparons aussi le rendu avec et sans la création d'un maillage unique birésolution. Cela permet de distinguer les gains et les pertes entre les deux approches.

Nos temps de rendu quand le maillage birésolution est utilisé sont similaires voire meilleurs aux dernières approches multirésolution. On peut de plus remarquer que :

- La cohérence temporelle accélère le temps d'affichage global quand le focus est rendu sans reconstruction de la zone de lien. Cette situation est bien adaptée à une exploration rapide des maillages. Ainsi, un déplacement interactif est garanti pour localiser les zones remarquables.
- La construction et la mise à jour du lien introduisent des surcoûts en étape I – comme la détermination des relations d'adjacence nécessaires à l'ordre de visibilité. Ce surcoût est compensé par l'utilisation de la cohérence temporelle ce qui permet d'obtenir un temps de rendu global équivalent à l'approche originale.

Ce type de rendu (rendu volumique direct ordonné) est ainsi interactif si la zone d'intérêt conserve une taille raisonnable par rapport à la taille de l'ensemble de données, ce qui semble une assertion plausible au vu de la localité d'une grande partie des phénomènes (comme évoqué au chapitre 1, section 5.3). Une zone d'intérêt dont le diamètre égale 10 à 20% de la diagonale de la boîte englobante du maillage semble un bon choix garantissant un taux de rafraîchissement de 3 à 9 images par seconde.

Bien que ces améliorations aient permis d'accélérer sensiblement le temps de rendu pour une telle approche – car rappelons-le, le rendu volumique reste une des techniques de visualisation les plus coûteuses pour les grilles irrégulières ; la nécessité de calculer les relations d'adjacence dynamiquement est assez coûteuse pour contrecarrer les gains apportés par la cohérence temporelle.

Suite à de telles limitations, et toujours dans l'optique d'adapter les techniques de rendu volumique aux maillages à connectivité changeante dynamiquement (ou multirésolution plus généralement), nous proposons dans la section suivante une autre approche ne reposant plus sur les relations d'adjacence adaptée pour l'extraction birésolution réalisée sur carte graphique.

3 Méthode Projective sur Carte Graphique

Dans cette section, nous proposons un algorithme de rendu volumique direct ordonné implanté sur carte graphique adapté aux schémas multirésolution. Nous exposons dans un premier temps la problématique au sein de laquelle s'inscrit cet algorithme (section 3.1). Nous détaillons ensuite la mise en œuvre d'un tel algorithme (section 3.2). Enfin, nous analysons et discutons les résultats obtenus (section 3.3).

3.1 Problématique

Nous nous plaçons dans le cadre où le maillage birésolution est extrait au sein de la carte graphique et nous souhaitons réaliser une exploration interactive de cet ensemble de données en utilisant un rendu volumique direct ordonné.

À la fin du calcul de l'extraction au sein du processeur de primitives programmable, le maillage est contenu dans un tampon de la mémoire graphique (*transform feedback buffer*) sans avoir subi l'étape de tramage. Il est ainsi prêt à être traité directement comme données d'entrée par un algorithme de rendu.

Nous souhaitons répondre à la problématique suivante : réaliser un rendu volumique direct ordonné d'un maillage dont la fréquence de mise-à-jour de sa géométrie et de sa connectivité peut être égale à son nombre d'affichage.

Deux solutions sont envisageables pour y répondre (si on exclut les solutions trop lentes reposant sur le paradigme du balayage) : soit une implantation reposant sur un lancer de rayons, soit une méthode de projection.

Dans le premier cas, la transposition d'un tel algorithme sur carte graphique nécessite la transformation préalable du maillage irrégulier en un certain nombre de texture bi- et tridimensionnelles [WMKE04, BPCS06]. Pour les obtenir, cela implique soit de modifier l'algorithme sur carte graphique d'extraction du maillage birésolution en introduisant plusieurs rendus hors-écran grâce à l'écriture de *fragment shaders* ; soit d'effectuer cette transformation au sein du processeur central ce qui sera coûteux en rapatriement des données. Dans les deux cas, les solutions semblent peu efficientes, d'autant plus que le maillage peut être modifié entre chaque rendu. Nous écartons ainsi cette solution.

Les méthodes par projection semblent être les meilleures candidates. Afin que l'utilisateur puisse réaliser son exploration dans les meilleures conditions, celles-ci doivent être temps-réel ou au moins interactive (cf. chapitre 1, section 3.2). L'implantation de telles méthodes sur carte graphique semble être une solution pour atteindre cet objectif.

Néanmoins, parmi les techniques existantes basées sur la projection des tétraèdres, aucune de celles-ci, à notre connaissance, ne gère de façon efficace la mise-à-jour dynamique de maillages tétraédriques de manière indépendante. En effet, les méthodes existantes de multirésolution dans le cadre d'un rendu volumique direct sont dépendantes de l'algorithme d'affichage et ne sont pas adaptées pour les schémas multirésolution réalisés en amont de la phase de rendu [FMSW00, CCS05, ACS+07].

HAVS [CICS05], par exemple, l'une des techniques de rendu volumique direct les plus rapides à l'heure actuelle, impose un premier tri des faces au sein du processeur principal – c'est-à-dire en dehors de la carte graphique. Cette solution considère que le maillage est statique. Ainsi, celui-ci est envoyé sous forme de points géométriques (*vertex buffer*) à l'initialisation et cette information n'est jamais mise à jour. Lorsque le point de vue est modifié, seules les faces sont triées en fonction de la nouvelle direction du regard de l'observateur puis envoyées dans cet ordre avec un tampon d'indices (*index buffer*) à la carte graphique. Cela implique que le maillage est à la fois stocké sur la carte graphique mais aussi au sein de la mémoire centrale.

Si l'on adapte *HAVS* à notre schéma birésolution, cela impose que l'ensemble des faces géométriques (en d'autres termes la soupe de triangles composant les faces du maillage) doit être dans un premier temps rapatrié

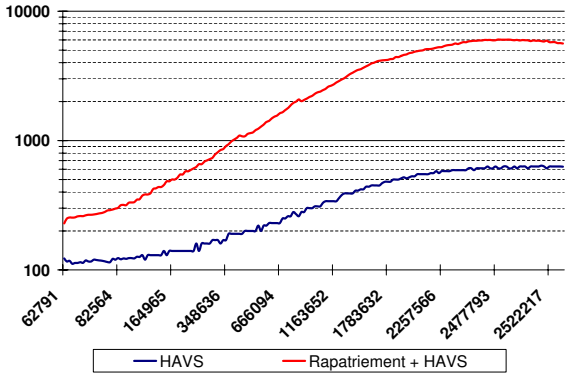


FIG. 4.15: **Comparaison entre HAVS pour un maillage statique et dynamique**. Le nombre de faces rendues est indiqué en abscisse, le temps en millisecondes en ordonné. La courbe bleue représente les temps de rendu pour une utilisation *classique* de HAVS sans mise à jour de la géométrie sur la carte graphique. Avec notre schéma birésolution construit au sein du processeur de primitives, les faces doivent être préalablement rapatriées au sein du processeur principal avant de pourvoir être traitées. La courbe rouge représente les temps de rendu dans le cadre du schéma birésolution.

au sein du processeur central afin d'y déterminer un premier ordre de visibilité, puis dans un second temps, renvoyé à la carte graphique selon ce nouvel ordre. Cela implique six fois plus de données à transférer par rapport à l'algorithme initial, sans prendre en compte la nécessité de synchronisation et la non-optimisation du transfert pour le rapatriement de l'information. Le surcoût imposé afin de greffer un tel système de rendu au sein de notre schéma birésolution est donné au sein de la figure 4.15. Dans le pire des cas, l'intégration de *HAVS* au sein de notre schéma d'extraction entraîne un temps de rendu dix fois supérieur à une utilisation de ce système avec un maillage statique.

Une telle solution n'est donc pas envisageable pour intégrer le rendu volumique direct à notre schéma birésolution (et plus largement aux techniques de multirésolution déjà développées). Nous proposons ainsi ci-après notre propre implantation afin de répondre à une telle problématique.

3.2 Rendu Volumique Direct

On suppose que le maillage est stocké au sein d'un tampon de la carte graphique (*transform feedback buffer*) sous la forme d'une soupe de cellules et l'on souhaite utiliser ce tampon comme entrée (*vertex buffer*) d'un algorithme de rendu volumique direct implanté totalement sur la carte graphique afin d'éviter les rapatriements coûteux sur le processeur central.

Notre solution est présentée au sein de la figure 4.16. Elle inverse le schéma habituel des méthodes de projection consistant à déterminer en premier lieu un ordre de visibilité des tétraèdres puis à les projeter au sein de l'écran.

Nous proposons dans un premier temps une solution exacte (section 3.2.1) puis un ensemble d'approximation afin de réduire la complexité temporelle de l'algorithme exact (section 3.2.2).

3.2.1 Solution Exacte

Le rendu volumique direct que nous proposons inverse le tri et la projection. En effet, dans un premier temps, nous transformons chaque tétraèdre en sa projection de triangles au sein de l'espace image (c'est-à-dire que l'on conserve sa profondeur dans l'espace objet) *via* l'utilisation du processeur de primitives comme nous l'avons détaillé dans la section 2.3. En sortie, une soupe de triangles est obtenue qui peut ainsi être triée selon le point de vue actuel puis accumulée. Cette soupe de triangles est stockée au sein d'un tampon de géométrie (*transform feedback buffer*).

Puis, dans un second temps, l'ordre de visibilité est assuré grâce à un algorithme de *depth-peeling* qui permet d'accumuler au fur et à mesure les contributions de chaque triangle. L'intégration se réalise donc de

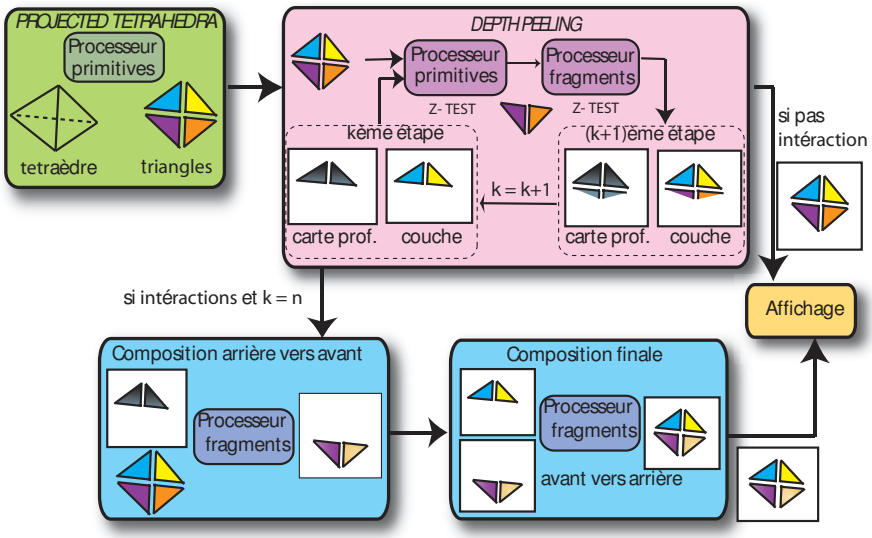


FIG. 4.16: **Tri et méthode projective sur carte graphique**. Le rendu volumique est décomposé en plusieurs étapes indépendantes. 1. Projection des tétraèdres en triangles en utilisant *Projected Tetrahedra*. 2. Tri et composition en utilisant le *depth-peeling* permettant à chaque étape d'extraire un tampon (*z-buffer*) et une couche (*layer*) de profondeur. Des approximations sont réalisées en utilisant un test de profondeur (*z-test*) au sein du processeur de primitives ou en stoppant l'extraction des couches à un certain niveau n et en générant pour les primitives restantes un tri non ordonné.

l'avant vers l'arrière (cf. définition 4.7). Notre algorithme de tri et de composition se déroule ainsi en plusieurs étapes de rendu, chacune appliquant des *shaders* différents.

1. **Initialisation** :
   a. *Depth-Peeling* : Rendu de la scène dans une texture $l_0 = (r_0, g_0, b_0, a_0)$ avec initialisation du tampon de profondeur z_0 ;
   b. *Composition* : Récupération du rendu hors-écran l_0 afin de composer la première contribution $c_0 = (a_0.r_0, a_0.g_0, a_0.b_0, a_0)$.
2. **Boucle Principale** : tant qu'il y a des fragments projetés à l'écran (*occlusion queries*),
   a *Depth-Peeling* : Extraction de la couche l_k en fonction du tampon de profondeur z_{k-1} ;
   b *Composition* : Accumulation avant vers arrière afin d'obtenir c_k grâce à l_k et l_{k-1} tel que : $c_k = l_{k-1} + (1 - a_{k-1})(a_k.r_k, a_k.g_k, a_k.b_k, a_k)$.

Une telle solution permet ainsi de réaliser un rendu volumique direct ordonné entièrement sur la carte graphique sans qu'aucune information géométrique ou de connectivité ne soit rapatriée au sein du processeur central. Néanmoins, la complexité d'une telle approche repose sur deux goulots d'étranglement. La profondeur du maillage, autrement dit le nombre de couches extraites lors du *depth-peeling*, est le premier. Ce nombre peut se révéler très important et nuire à l'interactivité du rendu lors de l'exploration. Le second est le nombre de fragments transmis à chaque étape du *depth-peeling*. En effet, les fragments sont rejetés uniquement après traitement. Ainsi, le nombre de fragments traités à chaque étape est toujours égal au nombre initial de fragments issus des primitives géométriques alors que le nombre de fragments rejetés augmente à chaque étape. Ce nombre important de fragments peut ainsi introduire une latence supplémentaire au sein de la carte graphique.

3.2.2 Solutions Approximantes

Afin de réduire la complexité temporelle de cet algorithme, nous proposons un ensemble de solutions approximantes lorsque l'utilisateur interagit avec la caméra ou la zone d'intérêt. En l'absence d'interactions, le rendu volumique peut être plus long et les étapes de rendu peuvent être réalisées dans leur intégrité.

L'utilisation du processeur de primitives au sein de l'étape de *depth-peeling* permet de résoudre en partie le goulot d'étranglement représenté par les fragments. En effet, le processeur de primitives peut tester en étape

k la position spatiale des triangles par rapport au tampon de profondeur z_{k-1}[6] de l'étape précédente. Il peut ainsi décider s'il envoie les primitives à la phase de tramage ou non. Pour cela, le test est réalisé par rapport à la position géométrique courante des sommets au sein de l'espace image. Si les trois sommets du triangle sont devant la profondeur z_{k-1} qui leur est associée, on peut alors valider le rejet d'une telle primitive car les fragments associés ne participeront pas au *layer* l_k extrait lors de cette étape.

Le rejet des primitives doit être fait avec précaution afin de ne pas trop détériorer l'image. Comme l'algorithme compose les triangles et non les tétraèdres, que la carte graphique est sujette à quelques imprécisions de calculs (en 32 bits) qui peuvent rapidement dégrader un rendu, et que le maillage n'est pas supposé être convexe ou connexe, ce test n'est effectué qu'à un certain nombre i d'itérations. En effet, rejeter directement un triangle dont les sommets sont devant le tampon de profondeur peut entraîner la disparition d'un triangle se trouvant en partie dans un espace vide (due à la non-convexité d'une zone spatiale) ou le long d'une arête déjà affichée (due aux imprécisions). Nous discutons le choix de i (nombre de pas d'itérations sans appliquer cette approximation) dans la section 3.3.1 suivante.

La seconde approximation possible concerne la méthode de composition. En effet, pour obtenir un rendu volumique direct ordonné correct, l'extraction de l'ensemble des couches est nécessaire. Lors de l'interaction, on peut supposer qu'une détérioration de l'image peut aussi être produite en arrière du volume sans que cela ne perturbe trop l'utilisateur. En effet, en accumulant les couleurs et les opacités de l'avant vers l'arrière, la contribution des fragments les plus éloignés de l'écran devient souvent de plus en plus négligeable. Ainsi, lorsque le nombre de couches atteint un certain n, le *depth-peeling* est stoppé et l'ensemble des triangles restant est composé dans un ordre quelconque.

La boucle principale est modifiée de la manière suivante lors d'une interaction de l'utilisateur :

2. **Boucle Principale** : tant que $\mathbf{k} \leq \mathbf{n}$,
   a. *Depth-Peeling* : Extraction de la couche l_k en fonction du tampon de profondeur z_{k-1} ;
   b. *Composition* : Accumulation avant vers arrière afin d'obtenir c_k grâce à l_k et l_{k-1} tel que : $c_k = l_{k-1} + (1 - a_{k-1})(a_k.r_k, a_k.g_k, a_k.b_k, a_k)$.
3. **Composition sans ordre** des triangles restants de l'arrière vers l'avant au sein de c_w.
4. **Composition finale** entre c_n et c_w via une composition de l'avant vers l'arrière.

L'évaluation d'une telle approximation est aussi discutée dans la section 3.3.1 suivante.

3.3 Résultats et Discussion

Nous présentons et discutons les résultats obtenus pour cette méthode ainsi que pour les approximations proposées lors de l'interaction de l'utilisateur (section 3.3.1). Ce rendu a ensuite été intégré au sein de notre approche birésolution en mémoire externe implantée sur la carte graphique (section 3.3.2). Enfin, nous discutons cette méthode (section 3.3.3). Quelques résultats visuels sont fournis au sein de la figure 4.17.

3.3.1 Évaluation des approximations

Nous effectuons tout d'abord une évaluation des différentes approximations que nous avons proposées : tri d'un nombre limité de couches avec accumulation des couches restantes sans ordre (évaluation de n), et utilisation d'un test de profondeur pour les triangles afin de les rejeter pour limiter le nombre de fragments (évaluation de i).

Pour chaque type d'approximation, on évalue à la fois leurs répercutions sur la complexité temporelle et la dégradation des images. Afin d'évaluer la dégradation des rendus par rapport à un rendu exact, une erreur ε est définie.

Définition 4.9 Erreur dans l'espace image ε : Soient deux images I et J, on définit l'erreur normalisée ε_{ij} entre les pixels $I_{ij} = (r^I_{ij}, v^I_{ij}, g^I_{ij}, a^I_{ij})$ et $J_{ij} = (r^J_{ij}, v^J_{ij}, g^J_{ij}, a^J_{ij})$ telle que :

$$\varepsilon_{ij} = \sqrt{\frac{1}{4}(r^I_{ij} - r^J_{ij})^2 + \frac{1}{4}(g^I_{ij} - g^J_{ij})^2 + \frac{1}{4}(b^I_{ij} - b^J_{ij})^2 + \frac{1}{4}(a^I_{ij} - a^J_{ij})^2} \in [0,1]$$

[6] avec toujours la convention que le pixel le plus proche est à une profondeur de 1 et que le plus éloigné de 0.

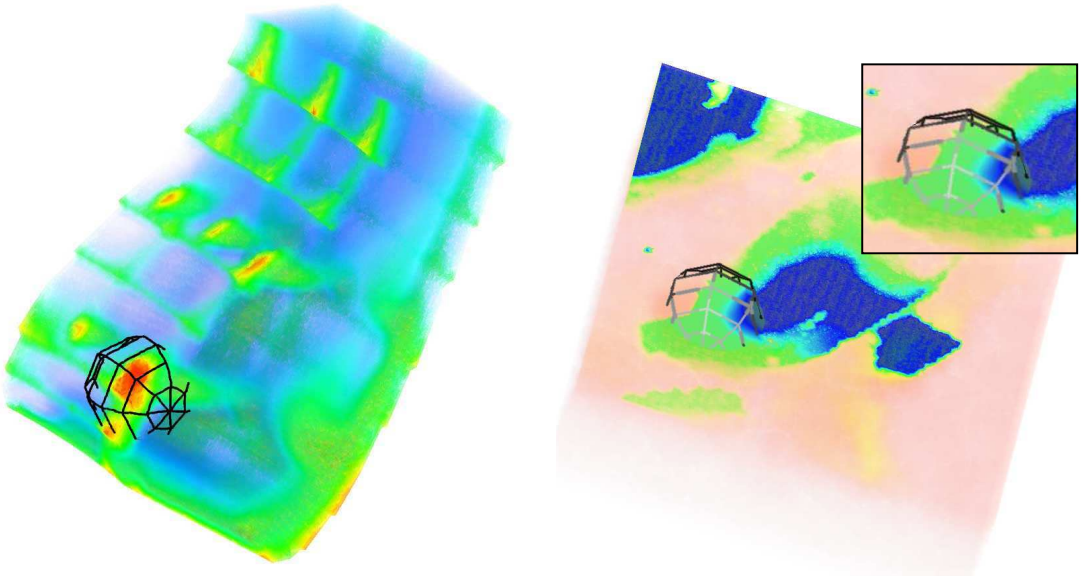


FIG. 4.17: **Exemples de rendu volumique direct avec notre méthode sur carte graphique**. Le rendu volumique direct ordonné est intégré au sein de notre schéma d'extraction d'un maillage birésolution. Ces rendus sont effectués respectivement sur l'ensemble *Comb(r)* et *Sf1*. La boule d'intérêt est représentée en filaire.

avec $I_{ij}, J_{ij} \in [0,1]^4$.

Le protocole afin d'évaluer la dégradation est le suivant. Le point de vue est fixe. On effectue dans un premier temps un rendu exact puis un rendu approximatif. Les deux images sont ensuite comparées à l'aide de l'erreur ε afin d'évaluer les similitudes et les disparités mais uniquement sur les pixels porteurs d'information. Cela signifie que les pixels équivalents à la couleur de fond sont rejetés de l'évaluation.

Ce protocole permet ainsi de construire un histogramme des erreurs ε_{ij} échantillonnées sur un pas de $0,01$ afin d'obtenir cent valeurs. On obtient ainsi pour chaque erreur $\varepsilon_{ij} = 0; 0,01; 0,02; \ldots; 1$ une évaluation du pourcentage de pixels porteurs de sens ayant cette erreur. La hauteur du premier bâton permet ainsi d'évaluer simplement le nombre de pixels identiques et la courbe de l'histogramme dans son ensemble la dégradation de l'image. Pour des raisons de clarté l'histogramme sera représenté au sein des figures suivantes sous deux formes. Une première sous forme de bâtons incluant le nombre de pixels avec une erreur nulle et limitée aux vingt premières erreurs environ (de $\varepsilon_{ij} = 0$ à $\varepsilon_{ij} = 0,2$) et une seconde sous la forme de courbes excluant les pixels d'erreur nulle pour des raisons de visibilité.

Nombre limité de couches correctement triées On évalue dans un premier temps l'influence du nombre limité *n* de couches correctement triées (de l'avant vers l'arrière) au sein du rendu volumique. L'évolution de la complexité temporelle en fonction du nombre de couches est donnée au sein du tableau 4.4 pour trois ensembles de données représentatifs. Le nombre de couches est diminué en tant que pourcentage par rapport au nombre total de couches nécessaires à l'intégration exacte du rendu volumique. Logiquement la complexité de l'algorithme se trouve fortement réduite lorsque le nombre *n* de couches exactement triées diminue fortement. On obtient ainsi un gain moyen de $1,83$ lors que le nombre *n* de couches est réduit de moitié et de $18,3$ lorsque celles-ci ne représente plus que 5% des couches nécessaires.

La dégradation d'une telle approximation est représentée au sein des graphiques de la figure 4.18 pour deux ensembles de données : *SPX* et *Torso*. La dégradation est calculée pour plusieurs valeurs de couches correctement triées (75%, 50%, 25%, 10% et 5% par rapport au nombre initial) et selon plusieurs points de vue différents. Plusieurs constatations peuvent être faites. Tout d'abord, comme on pouvait s'y attendre, la détérioration de l'image augmente au fur et à mesure que le nombre *n* de couches effectivement triées diminue. Néanmoins, le nombre de pixels ayant une erreur nulle reste dans tous les cas majoritaire à ceux ayant une erreur. Les erreurs sont ensuite réparties sous la forme d'une gaussienne se décalant vers les hautes valeurs au fur et à mesure de la permissivité du tri. Néanmoins, la quantité maximale de pixels pour l'erreur moyenne

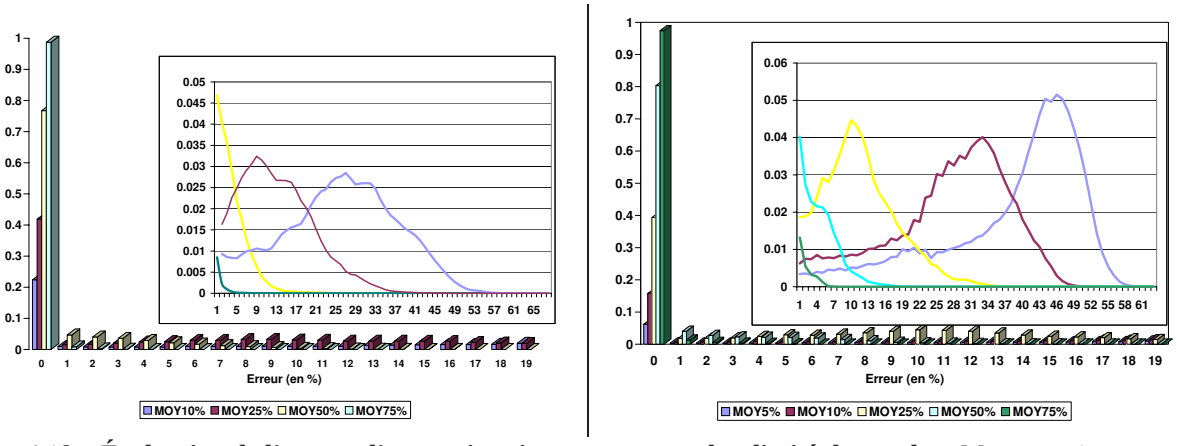


FIG. 4.18: **Évaluation de l'erreur d'approximation pour un nombre limité de couches**. Moyennes issues de plusieurs points de vue pour les ensembles *SPX* (à gauche) et *Torso* (à droite). On exprime les erreurs normalisées en pourcentage sur l'axe des abscisses et leur quantité en ordonnée. L'histogramme et les courbes représentent la même chose, les courbes étant à une échelle grossie afin de mettre en valeur les petites erreurs. Les couleurs sont attribuées en fonction du nombre de couches correctement triées par rapport au point de vue courant (renseignées en pourcentage). On note ainsi que plus le nombre de couches bien triées diminue plus le nombre d'erreurs augmente mais que celles-ci restent petites.

Données	100%	75%	50%	25%	10%	5%
SPX	446	350	240	126	60	–
BuckyBall	24 639	21 284	14 425	6 855	1 591	1 244
Torso	15 965	12 109	8 293	4 053	1 724	951

TAB. 4.4: **Temps du rendu en fonction d'un nombre limité de couches**. Moyennes temporelles en millisecondes issues de plusieurs points de vue pour les ensembles *SPX*, *BuckyBall* et *Torso*. Chaque pourcentage représente le nombre de couches conservées correctement triées.

ne dépasse jamais les 5% de la globalité des pixels. On a donc plus de petites erreurs réparties que de pixels isolés avec de grandes erreurs. De plus, lorsque le nombre de couches exactement triées est proche du nombre initial, les erreurs sont restreintes et centrées sur des petites valeurs. Dans des cas extrêmes (5-10%), les erreurs ne dépassent pas $0,6$ ce qui veut dire que la variation des canaux *RGBA* est restreinte et ne produit pas des variations abbérantes qui pourraient fausser l'interprétation au cours de l'interaction.

Faisons enfin remarquer que la dégradation obtenue pour 50% des couches uniquement triées est équivalente pour l'ensemble de données *Torso* à celle obtenue par l'algorithme *HAVS* pour un k-buffer de taille $k = 6$ [CICS05].

Rejet des primitives On évalue dans un second temps le rejet des triangles au sein du processeur de primitives grâce à un test de profondeur. Pour cela, on a tout d'abord concentré notre évaluation sur un ensemble de données : *SPX*. Nous avons fait varier le seuil i à partir duquel on applique le test de profondeur. Par exemple, si $i = 42$ on effectue toutes les 42 itérations une mise à jour du rejet des primitives avec les cartes de profondeur correspondant respectivement à la première carte de profondeur pour l'itération $i = 42$, à la quarante-deuxième pour l'itération $i = 84$, etc...

Les erreurs liées à l'approximation sont exposées au sein de la figure 4.19. Cette fois-ci, on remarque que l'utilisation du rejet de primitives modifie peu l'image finale même si l'on devient assez permissif avec cette condition ($i = 5$ par exemple). En effet, dans toutes les situations plus de 80% des pixels ont une erreur nulle et si celle-ci ne l'est pas, l'erreur ne dépasse pas 10%, la seule chose changeant réellement étant la hauteur de la gaussienne d'erreur pour $\varepsilon = 0,01$ qui augmente. Ainsi l'erreur reste dans sa globalité minime et dans le cadre de cette approximation ne crée pas de détériorations de l'image pouvant perturber l'utilisateur.

Les temps moyens pour *SPX* – calculés à partir de plusieurs points de vue – sont respectivement de 450 *ms* sans rejet au sein du processeur de primitives, 331 *ms* si rejet toutes les 42 couches, 326 *ms* si 21, 300 *ms* si 10 et 285 *ms* si 5. On remarque donc un gain non négligeable de $1,58$ lors de l'approximation la plus importante.

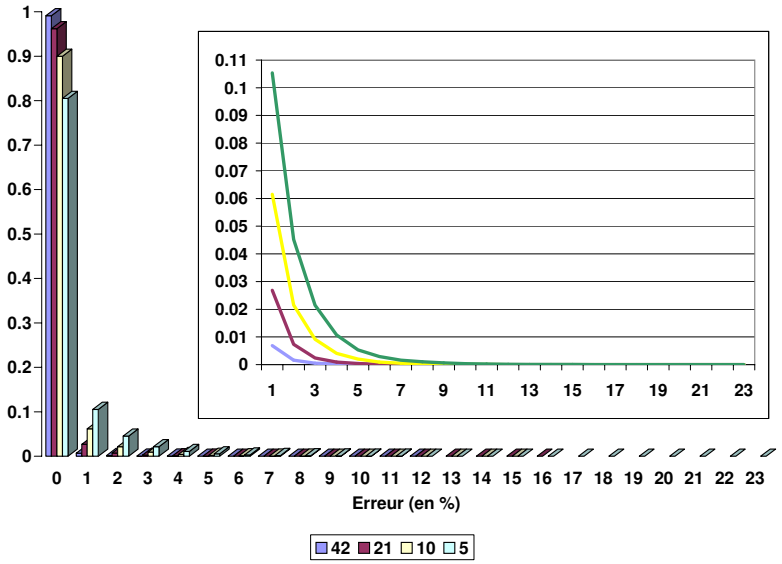


FIG. 4.19: **Évaluation de l'erreur d'approximation suite à un rejet de primitives**. Moyennes issues de plusieurs points de vue pour l'ensemble *SPX*. On exprime les erreurs normalisées en pourcentage sur l'axe des abscisses et leur quantité en ordonnée. L'histogramme et les courbes représentent la même chose, les courbes étant à une échelle grossie afin de mettre en valeur les petites erreurs. Les couleurs sont attribuées en fonction de l'appel au rejet des primitives toutes les *i* fois. On note ainsi que le rejet de primitives détériore peu le rendu final.

Données	moy_0	min_0	max_0	erreur max.	temps initial	temps obtenu
SPX	99,1	98,2	99,9	14	423,3	339,2
Blunt	95,1	76,7	100	18	2 634,8	1 873,5
Post	78,2	51,1	97,7	14	12 759	5 817,5
Torso	99,2	98,2	99,9	6	16 597	12 006

TAB. 4.5: **Évaluation quantitative et temporelle du rejet de primitives**. Pourcentages de pixels minimum min_0, en moyenne moy_0 et maximum max_0 ayant une erreur nulle extraits à partir d'un certain nombre de points de vue représentatifs, erreur maximale (en pourcentage) atteint par au moins un pixel de l'image ainsi que les temps sans et avec l'approximation reposant sur le rejet des primitives pour les ensembles de données *SPX*, *Blunt*, *Post* et *Torso*.

Suite à la constatation que l'image était peu détériorée lors de l'utilisation d'une telle approximation, nous avons validé celle-ci pour un ensemble de maillages tétraédriques représentatifs donnés au sein du tableau 4.5. Nous avons porté notre choix, après plusieurs tests, pour une valeur de *i* égale au maximum de tétraèdres se trouvant autour d'un sommet. Au sein de ce tableau, sont renseignés le pourcentage de pixels en moyenne (minimum et maximum) ayant une erreur nulle extraits à partir d'un certain nombre de points de vue représentatifs, l'erreur maximale (en pourcentage) atteint par au moins un pixel de l'image ainsi que les temps sans et avec l'approximation. On remarque ainsi qu'un gain de 1,56 est obtenu en moyenne – d'autant plus important que l'image est détériorée ; avec un nombre moyen de pixels sans erreur s'élevant à 92,9% de l'image.

Discussion L'ensemble de ces évaluations quantitatives permet d'affirmer que les erreurs générées par de telles approximations réduisent la qualité de l'image proportionnellement à leur permissivité mais ne produisent pas d'artefacts incohérents par rapport au rendu initial. Cela permet ainsi en fonction des besoins de l'utilisateur et du maillage exploré d'adapter ces approximations afin de garantir un compromis entre la qualité du rendu et l'interactivité de l'exploration. Des approximations plus grossières permettront une exploitation interactive des données au détriment de la précision du rendu mais la garantie que les premières couches soient correctement triées limitent une variation visuelle trop importante comme nous avons pu le constater en pratique. Néanmoins, cette étude pourrait être couplée d'une évaluation qualitative sur l'impact réel de telles approximations au

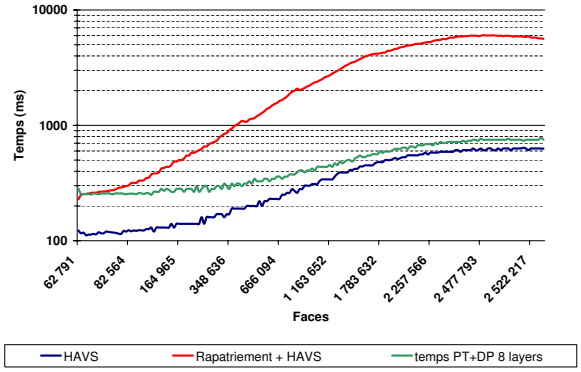


FIG. 4.20: **Comparaison de notre méthode avec HAVS**. Ces temps sont des moyennes extraites durant le grossissement de la zone d'intérêt au sein d'un maillage. Les trois courbes représentent respectivement : le temps de HAVS pour un maillage statique (bleu), le temps de HAVS adapté à notre maillage birésolution (rouge) et le temps de notre méthode avec un nombre de couches correctement triées égal à cinq pour cent du nombre total moyen de couches et le rejet des primitives selon le nombre de tétraèdres maximal entourant un sommet.

niveau perceptif de l'utilisateur au cours de son exploration mais nous n'avons pas eu le temps de la mettre en place dans le cadre de ce doctorat.

Par la suite, basé sur notre évaluation quantitative et notre propre perception du rendu volumique, nous avons décidé de fixer le nombre i au nombre maximal de tétraèdres entourant un sommet et n à cinq pour cent des couches nécessaires à un tri de visibilité exact. De telles approximations comme nous l'avons vu détériorent en partie l'image mais garantissent une interactivité lors de l'exploration ou du changement de point de vue suffisante pour le confort de l'utilisateur.

3.3.2 Intégration dans le schéma birésolution

Nous avons intégré un tel rendu dynamique au sein de notre schéma birésolution. Notre algorithme utilise directement le tampon de géométrie en entrée et produit l'affichage final du rendu volumique direct en utilisant les approximations définies et discutées précédemment.

Nous avons dans un premier temps comparé la complexité de notre approche approximante avec celle du système *HAVS*, qui rappelons-le, bien qu'un des algorithmes les plus performants pour réaliser un tel rendu ne l'est pas pour gérer les modifications dynamiques d'un maillage tétraédrique extrait sur carte graphique. Les résultats sont présentés au sein de la figure 4.20. La complexité de notre approche (avec n et i fixés) tend à être égale à celle d'affichage de *HAVS* lorsque le nombre de triangles à traiter augmente. Cela est un point intéressant puisque nous avons principalement pensé cet algorithme pour qu'il puisse afficher efficacement de gros maillages de données. Notons qu'ainsi cette approche est performante par rapport à l'intégration directe de *HAVS* nécessitant, comme nous l'avons déjà évoqué, le rapatriement de l'information géométrique au sein du processeur principal avant de pouvoir effectuer le rendu.

Les temps globaux du rendu dans le cadre de son intégration au sein du schéma birésolution en mémoire externe sur carte graphique sont donnés au sein de la figure 4.21. Ces graphes sont issus d'une exploration type sur le maillage *Sf1* permettant d'étudier la superposition des différentes couches géologiques là où celles-ci sont densément proches non loin de la surface (cf. figure 4.17 droite). Ils mettent en évidence à la fois les temps d'affichage mais aussi les mécanismes de chargement liés à notre approche en mémoire externe. Le graphe de gauche représente l'exploration de *Sf1* au sein d'un découpage spatial de $5 \times 5 \times 4$ blocs, celui de droite au sein de $6 \times 6 \times 6$ blocs. Ces deux graphes permettent ainsi de simuler différents comportements en fonction des limitations architecturales des ordinateurs pour la même exploration.

Les histogrammes représentent le nombre de fichiers nécessaires à l'extraction du maillage birésolution. On a supposé qu'uniquement huit fichiers pouvaient être chargeables en mémoire. Ainsi, dans un cas (celui de gauche) l'ensemble des blocs sont chargeables en mémoire alors que cela n'est pas le cas dans l'autre (à droite). On peut ainsi remarquer le surcoût des chargements des fichiers intervenir dans la courbe représentant le temps global.

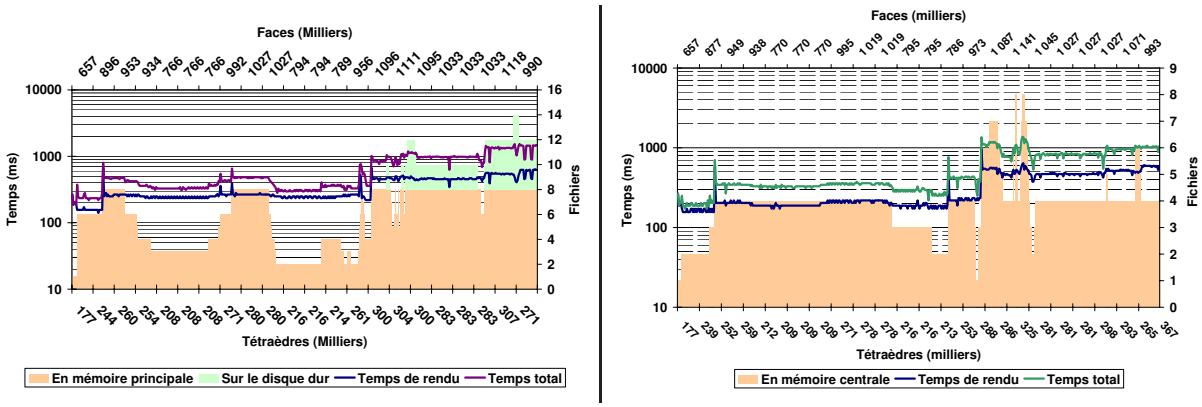


FIG. 4.21: **Intégration du rendu volumique direct au sein du schéma birésolution**. Ces courbes sont obtenues lors de l'exploration du maillage *Sf1*. Les histogrammes représentent le nombre de fichiers chargés au sein de la mémoire centrale ou chargés depuis le disque dur. Les courbes représentent le temps de rendu (en bleu) et le temps total avec l'extraction et le chargement (en violet). Le nombre de couches correctement triées étant cinq pour cent du nombre total moyen de couches et le rejet des primitives étant activé.

Le temps de rendu de notre approche suite aux choix que nous avons fixés est ainsi *interactif* (cf. chapitre 1, section 3.2) tout au long de l'exploration et l'ajout du chargement des données et de l'extraction ne pénalise pas cette constatation excepté lorsque la place mémoire n'est plus assez importante pour garantir la conservation des données. Le surcoût des entrées/sorties réduit alors le taux de rafraîchissement a une image par seconde. Rappelons que notre structure en mémoire externe n'a pas été optimisée comme nous l'avons discuté longuement au sein du chapitre précédent.

3.3.3 Bilan et Discussion

Une telle solution permet ainsi d'obtenir une exploration interactive mélangeant l'extraction d'un maillage birésolution sur carte graphique et un rendu volumique direct ordonné approximé lorsque les conditions matérielles permettent de charger l'ensemble des fichiers nécessaires au sein de la mémoire centrale.

Néanmoins, la solution proposée peut subir quelques améliorations. Le *depth-peeling* n'est à l'heure actuelle pas optimisé afin de garantir les meilleures performances. Une approche *duale* [BM08] permettant d'extraire les couches par paires permettrait ainsi d'améliorer les performances et de diminuer en même temps les erreurs dues au tri non ordonné sur les couches restantes. Une autre solution pourrait être d'utiliser l'algorithme de tri proposé dans [GHLM05] sur les triangles afin de générer un nouveau buffer ordonné qui pourrait alors être utilisé comme entrée de *HAVS*. Une telle solution demanderait là encore une complète évaluation afin de déterminer les erreurs générées par le *k*-buffer.

A contrario, l'arrivée prochaine de nouvelles cartes graphiques regroupées par paire[7] (voire plus) permettrait sans aucun doute d'améliorer les performances d'une telle approche soit en partageant l'écran en deux soit en effectuant de manière successive les rendus sur l'une puis l'autre carte graphique ce qui respectivement diminuerait le nombre de fragments traités ou augmenterait le temps disponible pour la réalisation des calculs afin d'améliorer dans les deux cas le taux de rafraîchissement final.

4 Autres Techniques de Visualisation

Nous détaillons dans cette section, l'intégration des autres techniques usuelles : rendu par points, plans de coupe, isosurfaces ; au sein de notre schéma birésolution à la fois pour sa version sur processeur principal et sur carte graphique. Un échantillon de résultats visuels est disponible au sein de la figure 4.22. L'utilisateur peut ainsi mélanger plusieurs techniques de visualisation entre elles afin de l'aider dans sa phase d'exploitation à comprendre la simulation et la localité des phénomènes.

[7] comme les nVidia GeForce GTX 295 ou la ATI FirePro 2450.

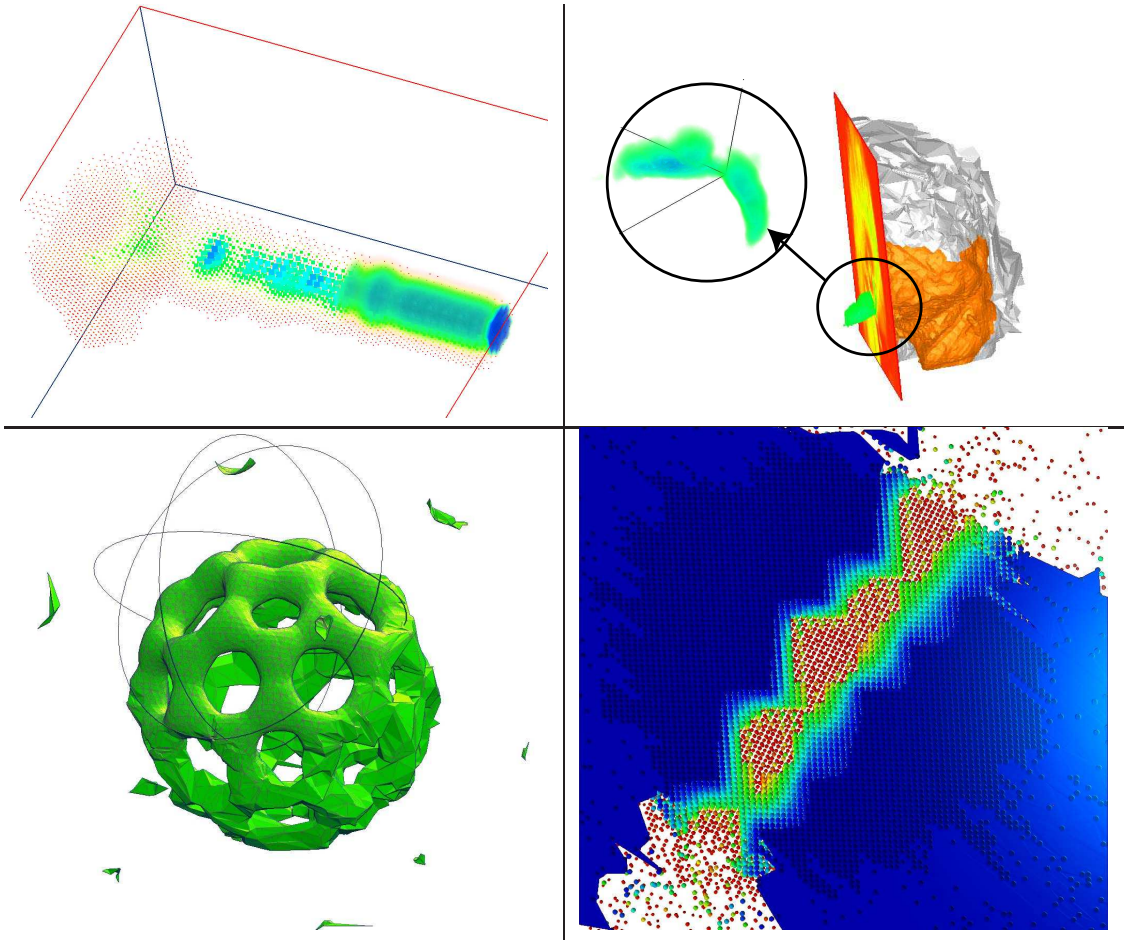


FIG. 4.22: **Techniques de visualisation usuelles intégrées au sein du schéma birésolution**. En haut : au sein de la version CPU pour les ensembles de données *Fuel* et *DTI*. La zone d'intérêt est déplacée le long de l'injection pour *Fuel*. Un rendu volumique direct est réalisé en son sein alors qu'un plan de coupe et un rendu par points assure une représentation minimaliste mais porteuse de sens pour le maillage grossier. Pour *DTI*, la zone d'intérêt est centrée sur l'hypothalamus. Un plan de coupe texturé ainsi que l'extraction d'une surface isovaleur (incluse en partie dans la zone d'intérêt : partie orangée) assure une mise en relation de la zone d'intérêt par rapport au volume médical complet. En bas : au sein de la version GPU pour les ensembles *BuckyBall* et *Sf1*. Une extraction birésolution d'une surface isovaleur est réalisée pour le premier, alors qu'une isosurface couplée à un rendu par points permet de comprendre les relations entre les différentes couches géologiques du second.

L'intégration du rendu par points (ou *point sprites*) et des plans de coupe est faite comme pour des maillages tétraédriques monorésolution. Nous ne détaillons donc pas ici l'implantation de telles techniques de visualisation. L'extraction de surfaces isovaleurs représente par contre un intérêt dans son intégration au sein du processeur principal afin de prendre en compte la cohérence temporelle. Nous précisons ainsi dans la suite de cette section l'intégration d'une telle technique au sein de notre schéma birésolution et les résultats obtenus en terme d'efficacité.

4.1 Intégration des Isosurfaces

L'extraction des isosurfaces est réalisée grâce à un *Marching Tetrahedra* [GH95] au sein du processeur central si l'extraction est réalisée au sein de celui-ci ou sur directement sur carte graphique [BCL06] si l'extraction est réalisée elle-même sur carte graphique.

Lorsque l'extraction des surfaces isovaleur est réalisée au sein du processeur central, la cohérence tempo-

relle peut être mise à contribution afin d'améliorer les taux d'extraction. Pour cela, l'isosurface grossière est extraite une seule fois tant que l'isovaleur n'est pas modifiée. À l'inverse, l'isosurface à haute résolution est mise-à-jour dès que la zone d'intérêt est déplacée par l'utilisateur.

L'extraction fine est réalisée sur les ensembles τ_c (équation (2.1)) de tétraèdres fins. On rappelle que τ_c contient l'ensemble des tétraèdres fins dont au moins un des sommets a pour image le sommet grossier c par la surjection $\wp$ (cf. chapitre 2, section 2.4). Dès qu'un sommet grossier c devient actif (cf. chapitre 3, section 1.2.2), l'ensemble τ_c est inséré au maillage birésolution. L'isosurface correspondant à cet ensemble est calculée pour l'ensemble des tétraèdres fins avec plusieurs calculs indépendants pour les tétraèdres de lien dont la forme géométrique dépend de la position spatiale de la boule d'intérêt. Ils peuvent en effet devenir actifs ou rester de lien mais selon différentes configurations (trois au plus). Cette solution permet d'éviter des calculs redondants lorsque la boule d'intérêt ne se déplace qu'au sein d'un unique cluster ou d'un ensemble constant de clusters.

Le maillage birésolution assure de plus l'extraction d'une seule et unique surface isovaleur à deux résolutions, l'union étant réalisée par les triangles issus des tétraèdres de lien. La création de cette unique isosurface permet ainsi d'éviter le recours à des astuces visuelles comme un mélange par transparence proposé par Cignoni *et al.* [CDFM+94].

4.2 Résultats et Discussion

Nous discutons l'insertion du rendu indirect par isosurfaces au sein du schéma birésolution, dans un premier temps au sein du processeur central (section 4.2.1), puis au sein de la carte graphique avec l'approche en mémoire externe (section 4.2.2).

4.2.1 Avec cohérence temporelle

Nous avons effectué un certain nombre de tests en utilisant plusieurs explorations au sein de différents ensembles de données et moyennés l'ensemble de ces résultats. Ces tests ont été réalisés avec et sans cohérence. Les débits moyens sont les suivants : 250 000 tétraèdres à la seconde sans cohérence et 650 000 tétraèdres à la seconde avec cohérence. L'utilisation de la cohérence temporelle au sein de l'extraction de l'isosurface permet ainsi en moyenne un gain temporel de $2,6$ par rapport à une version sans celle-ci.

Notons que contrairement au rendu volumique direct ordonné, l'extraction d'isosurface ne nécessite pas les relations d'adjacence. Il n'y a donc aucun surcoût dû à la construction du lien. L'exploitation de la cohérence temporelle est donc complète.

4.2.2 Au sein de la carte graphique

L'extraction des isosurfaces est réalisée au sein d'un processeur de primitives qui calcule en parallèle l'ensemble des triangles appartenant à la surface isovaleur courante. Ce traitement utilise en entrée le tampon créé par le schéma birésolution et contenant la soupe de cellules. L'extraction de l'isosurface est donc pleinement intégrée au sein de notre approche en mémoire externe.

Les graphes de la figure 4.23 représentent les temps de rendu globaux pour l'affichage du maillage birésolution grâce à des isosurfaces. Ces courbes sont obtenues comme moyenne d'un parcours d'exploration du maillage *Sf1*. On remarque ainsi que le temps d'extraction des isosurfaces est linéaire en fonction du nombre de primitives. La complexité moyenne de l'implantation sur carte graphique de l'extraction de l'isosurface permet de traiter environ neuf millions de tétraèdres à la seconde.

Le plus coûteux reste comme pour le rendu volumique direct ordonné le chargement en mémoire centrale des fichiers avant leur transmission à la carte graphique. Ainsi, le pipeline complet du chargement des données, en passant par l'extraction, jusqu'au rendu permet ainsi de traiter environ deux millions de tétraèdres à la seconde lorsque les données sont entièrement chargeables en mémoire centrale. Lorsque des chargements sont nécessaires, ce temps est diminué à un million de tétraèdres à la seconde lorsqu'un fichier doit être chargé à chaque rendu et cinq cent mille lorsque deux sont lus depuis le disque dur. Rappelons enfin que le processus de chargement n'est pas optimisé et que les débits de traitement peuvent ainsi être encore améliorés.

On a ainsi, pour ce rendu indirect, un temps d'extraction et de rendu interactif même lorsque la mémoire centrale est limitée et qu'un certain nombre de fichiers doit être lu depuis le disque dur à chaque rendu.

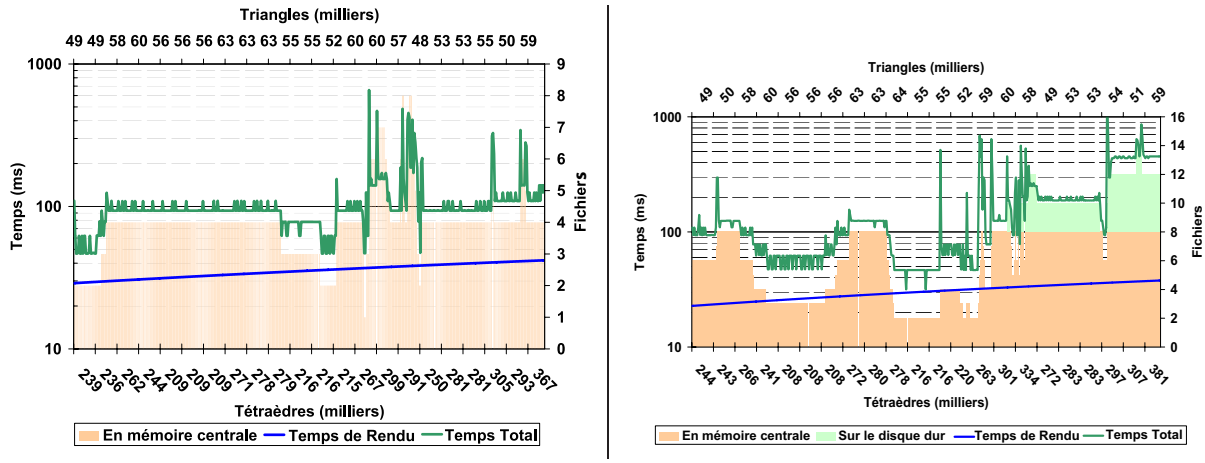


FIG. 4.23: **Intégration des isosurfaces au sein du schéma birésolution**. Ces courbes sont obtenues lors de l'exploration du maillage *Sf1*. Les histogrammes représentent le nombre de fichiers chargés au sein de la mémoire centrale ou chargés depuis le disque dur. Les courbes représentent le temps de rendu (en bleu) et le temps total avec l'extraction et le chargement (en vert).

4.2.3 Discussion

Une telle implantation permet ainsi dans les deux situations, en processeur central comme au sein de la carte graphique, d'obtenir une extraction d'isosurface interactive. Couplée avec un rendu par points (*point sprites*) ou un plan de coupe texturé, on peut ainsi obtenir une autre manière interactive d'explorer dans sa globalité une simulation sans utiliser le rendu volumique direct ordonné.

5 Bilan et Perspectives

Dans ce chapitre, nous avons proposé des algorithmes de rendu volumique direct ordonné adapté à des maillages dynamiques, c'est-à-dire à la géométrie et à la connectivité pouvant changer dynamiquement à chaque affichage. C'est à notre connaissance la première fois que la technique de visualisation du rendu volumique direct est intégré aux méthodes multirésolution, et non l'inverse.

Nous avons ainsi, dans un premier temps, adapté l'algorithme *SXMPVO* à notre schéma birésolution implanté au sein du processeur principal. Pour cela, nous avons inséré la cohérence temporelle au sein du tri des primitives lorsque le point de vue reste inchangé en remarquant que celui-ci est modifié de manière locale. Nous avons aussi proposé de nouvelles améliorations afin d'accélérer la totalité du pipeline grâce à une implantation efficace du *depth-peeling* sur carte graphique et de la projection des tétraèdres au sein du processeur de primitives.

Les modifications reposant sur la carte graphique ont permis d'accélérer avec un gain raisonnable la globalité du pipeline graphique. L'utilisation de la cohérence temporelle est quant à elle plus mitigée. Bien que celle-ci permette d'accélérer une partie de l'algorithme de manière significative, le calcul des relations d'adjacence entre les tétraèdres de lien réduit ce gain à la complexité initiale de *SXMPVO* pour un maillage monorésolution. L'union de ces deux modifications permet tout de même d'accélérer le rendu volumique direct par rapport à l'état de l'art reposant sur un tri exact des primitives.

Dans un second temps, nous avons proposé une nouvelle technique implantée dans sa totalité au sein de la carte graphique. Elle repose sur la projection des tétraèdres (*Projected Tetrahedra*) et un algorithme de tri dans l'espace image (*depth-peeling*).

La complexité de cette solution dépend fortement du nombre de couches à trier lors de l'étape de *depth-peeling* et celui-ci peut se révéler grand pour des maillages tétraédriques complexes. Afin de diminuer ce goulot d'étranglement, nous avons proposé deux types d'approximation : une cherchant à limiter le nombre de fragments traités au sein de la carte graphique grâce à l'ajout d'un test de profondeur sur les triangles au sein du processeur de primitives, l'autre limitant le nombre de couches exactement triées de l'avant vers l'arrière. Nous avons évalué ces deux approximations afin de déterminer l'ampleur de l'approximation par rapport au

gain temporel et ainsi sélectionner un compromis adapté dans le cadre de l'exploitation des données grâce à un rendu volumique direct ordonné.

Nous avons ensuite intégré cette technique au sein de notre schéma birésolution tout en assurant que les choix proposés permettaient d'égaler la puissance des meilleurs algorithmes de projection actuels comme *HAVS*. Cette technique grâce aux approximations permet ainsi de garantir une exploration interactive du maillage avec un rendu volumique direct ordonné.

Nous avons enfin pleinement intégré les autres techniques de visualisation indirectes pour les champs scalaires comme le rendu par points (*point sprites*), les plans de coupe texturés et les surfaces isovaleur.

Néanmoins, plusieurs perspectives sont envisageables concernant l'intégration des techniques de visualisation au sein de notre schéma birésolution. Dans un premier temps, de nombreuses améliorations peuvent encore être considérées concernant l'implantation du rendu volumique direct ordonné pour des maillages multirésolution. Les méthodes de projection semblent les plus adaptées pour s'intégrer dans de telles situations. Mais l'utilisation du *depth-peeling* pour réaliser le tri ne semble pas la solution la plus efficace. D'autres alternatives pourront donc être explorées comme des implantations de tri de primitives hors pipeline graphique (écrit en *CUDA* ou en *openCL*) lorsque le changement de contexte architectural entre le graphique *openGL* et le *calcul générique GPGPU* sera plus efficace au sein de la carte graphique. De plus, la possibilité d'une meilleure intégration du lancer de rayons au sein de la carte graphique[8] laisse présager de futures possibilités interactives quant à l'utilisation d'une telle technique pour des maillages multirésolution.

De plus, les techniques de visualisation concernant les champs vectoriels n'ont pas été intégrés au sein de notre schéma birésolution. Les méthodes directes (*hedgehog*) sont facilement implantables puisqu'elles sont identiques à des rendus par points. Par contre, le calcul de lignes de courant (*streamlines, pathlines*...) au sein de la carte graphique dans le cadre de notre schéma birésolution en mémoire externe représente un autre challenge intéressant, si l'on garde l'optique de ne pas calculer les relations d'adjacence afin de minimiser l'occupation mémoire.

Enfin, le développement de techniques parallèles au niveau du rendu suite à l'arrivée sur le marché de cartes graphiques couplées, ouvre de nouvelles perspectives pour les utilisateurs d'un ordinateur de bureau n'ayant pas d'accès à des grappes de calculs et de visualisation afin d'effectuer la validation de leurs simulations.

[8] avec l'annonce de *NVIDIA* de sa nouvelle API *NVIRT* pour *NVIDIA Ray-Tracing engine* reposant sur *CUDA*.

Conclusion

C'est lorsque nous croyons savoir quelque chose qu'il faut justement réfléchir un peu plus profondément.

Franck HERBERT - *Les Enfants de Dune*

Les travaux présentés au sein de ce mémoire ont été réalisés dans l'optique de garantir une visualisation interactive de grosses masses de données obtenues au sein de maillages tétraédriques. Nous avons ainsi répondu aux problématiques suivantes : la réduction du temps de prétraitement des données afin que celles-ci soient visualisables grâce à un ordinateur de bureau ; la réduction du temps d'extraction dynamique d'un maillage adaptatif au cours de l'exploitation ; et l'adaptation des techniques de visualisation à la modification dynamique des maillages tétraédriques. La résolution de telles problématiques permet ainsi d'obtenir une manipulation et une exploitation interactive des données.

Résumé des contributions

Dans un premier temps, nous avons réduit les temps de prétraitement en proposant un algorithme de simplification de maillages tétraédriques et la création d'une partition des sommets d'un maillage tétraédrique haute résolution contrainte par un maillage tétraédrique basse résolution.

La qualité des tétraèdres grossiers étant une nécessité pour diminuer les artefacts lors du rendu, nous avons étendu un algorithme existant qui décompose le maillage en deux parties distinctes : la surface de bord et l'intérieur, chaque partie subissant une simplification indépendante. Celles-ci sont ensuite réutilisées afin de réaliser une tétraédralisation contrainte de Delaunay. Le maillage grossier obtenu est ainsi simplifié plus drastiquement qu'avec les précédentes approches de simplification tout en conservant la topologie du champ scalaire associé et en assurant une certaine qualité des tétraèdres.

Les approches multirésolution usuelles nécessitent la mise en place d'algorithme glouton de simplification et de structures de données complexes afin d'extraire des relations de dépendances. Afin de s'en affranchir, nous avons ensuite proposé la construction d'une partition des sommets d'un maillage contrainte par une simplification de ce même maillage. L'élaboration d'une telle partition n'est assujettie à aucun algorithme de simplification. Elle vérifie de plus, avec un maillage grossier vérifiant certaines conditions géométriques, un certain nombre de propriétés permettant la construction dynamique d'un maillage unique birésolution, mélange de la résolution initiale avec sa simplification.

Ces algorithmes de prétraitement se sont révélés plus rapides que ceux utilisés jusqu'à présent pour la construction d'approches multirésolution et sont appliqués avec succès pour des maillages de plus de quarante

millions de tétraèdres avec une mémoire vive limitée à deux gigaoctets.

Nous avons ensuite proposé un nouveau schéma d'extraction de maillages adaptatifs reposant sur les besoins de l'utilisateur que nous avons nommé *BiRes*. Ce schéma extrait, à partir de la partition calculée lors des prétraitements, un maillage à deux résolutions : la résolution fine initiale et la résolution grossière qui a contraint la partition. Ce schéma d'extraction a été implanté au sein de la mémoire vive en exploitant la cohérence temporelle, au sein des nouvelles cartes graphiques possédant un processeur de géométrie, et enfin en mémoire externe afin de s'affranchir des limitations mémorielles de l'architecture. Ce schéma garantit des vitesses d'extraction d'un maillage birésolution qui n'ont jamais été atteints jusqu'alors par les précédentes approches multirésolution.

Nous avons enfin détaillé l'adaptation de techniques de visualisation usuelles pour des maillages tétraédriques dynamiques comme ceux extraits dans le cadre de notre schéma *BiRes*. Nous avons concentré nos efforts sur le rendu volumique direct implanté à la fois sur le processeur central et la carte graphique. La cohérence temporelle a été exploitée afin d'accélérer le tri des cellules en mémoire vive avant leur projection. Au sein de la carte graphique, nous avons proposé un nouveau pipeline de rendu avec un certain nombre d'approximations garantissant une exploration interactive. Nous avons aussi intégré et adapté les autres techniques de visualisation comme les isosurfaces, les plans de coupe texturés ou le rendu par points afin d'obtenir un logiciel complet proposant un panel de représentations utiles à l'exploitation des données dans le cadre du schéma *BiRes*.

De premiers retours industriels sur notre schéma birésolution *Bires* ont été recueillis. La volonté de simplifier et d'accélérer les prétraitements afin d'obtenir un maillage adaptatif interactif au sein de grosses masses de données a été perçue comme une proposition intéressante pouvant répondre pleinement aux attentes d'ingénieurs toujours lassés par le temps d'attente et les rendus difficilement contrôlables par un manque d'interactivité. Notre approche est ainsi en adéquation avec les problématiques actuelles et ouvre de nouvelles perspectives pour les défis auxquels seront confrontés les utilisateurs dans les années à venir.

Perspectives

La visualisation scientifique est un domaine qui se trouve sans cesse confronté à une évolution rapide à la fois des moyens architecturaux (supercalculateurs, cartes graphiques, ...) mais aussi des exigences d'interactivité et de précision (donc de traitements de grosses masses de données) des utilisateurs finaux. Cette évolution conduit à une diversification à la fois des simulations mais aussi des domaines d'applications. Afin de répondre à ces demandes de manière efficiente, la visualisation scientifique va devoir relever un certain nombre de défis dans les prochaines années.

Généricité des traitements et spécialisation des représentations Avec l'amélioration des techniques de discrétisation de l'espace pour créer des maillages tétraédriques ou hexaédriques mais aussi des scanners d'acquisition créant des nuages de points ou une succession ordonnée d'images bidimensionnelles, le "bestiaire" des données visualisables se diversifie un peu plus chaque année. La visualisation scientifique a jusqu'alors spécialisé ses techniques de prétraitement (simplification, multirésolution, segmentation, ...), de stockage (textures tridimensionnelles, structures topologiques, ...) et de rendu (principalement pour le rendu volumique direct) à des types de donnés spécifiques (grille régulière, maillage tétraédrique, nuages de points, ...). Elle se retrouve maintenant confrontée à des représentations de données hybrides mélangeant des maillages irréguliers à des voxels ou des nuages de points. On peut citer, par exemple, l'utilisation de maillages hybrides irréguliers (mélange de tétraèdres et d'hexaèdres) en simulation de flux ou l'utilisation d'un maillage irrégulier et de couches de points pour simuler la pénétration d'un corps dans des couches géologiques.

Pour répondre à cette diversification des discrétisations de l'espace au sein d'un même ensemble de données, la visualisation scientifique doit créer des méthodes génériques de traitement des données. Ces méthodes devront garantir une exploitation interactive des données. L'utilisateur ne sera alors plus contraint à mélanger (et donc implanter) des techniques spécialisées entre elles avec les risques que cela soit impossible ou trop lent pour l'exploitation des données. Cette généricité d'implantation est un défi important et complexe. Par exemple, pour les prétraitements (simplification, multirésolution, ...), les techniques élaborées reposent souvent sur les

propriétés géométriques et de connectivité du maillage (*octree* pour les grilles régulières, décimation locale pour les maillages irréguliers). Des approches plus simples – comme celle que nous avons tenté de proposer au sein de ce mémoire – s'affranchissant de telles contraintes sont une première piste pour obtenir de tels prétraitements génériques ; l'idéal étant de les supprimer lorsque cela est possible. De plus, l'élaboration de représentations génériques (rendus par points, glyphes, lignes, ...) passant outre les notions de connectivité et de formes de cellules (voxels, polyèdres, ...) permettrait la création d'un pipeline de traitement des données entièrement générique.

Néanmoins, la volonté d'une telle généricité des traitements ne doit en aucun cas restreindre les spécialisations des techniques de visualisation. Il est en effet important de constater que ces représentations répondent à un contexte scientifique et à des objectifs finaux spécifiques à la simulation. L'ingénieur se repose sur son expérience et ses habitudes pour choisir la représentation la plus apte à mettre en emphase un phénomène précis dans son contexte afin d'atteindre les objectifs recherchés. Il est ainsi usuel de réaliser un rendu volumique direct dans le domaine médical : typiquement les méthodes dites *X-Ray* et *MIP* ne sont pertinentes que pour certaines acquisitions. Les lignes de courant sont préférées pour représenter des simulations de flux. Ces solutions ne sont ainsi adaptées qu'à des situations précises. Dans d'autres circonstances, on peut préférer l'extraction d'une isosurface pour mettre en évidence la structure locale d'un champ plutôt qu'un rendu volumique direct diffus où la notion de profondeur peut être ambiguë. Il est aussi envisageable qu'aucune représentation actuelle ne permette de produire une image compréhensible pour certaines simulations.

La mise à disposition de supercalculateurs[9] à l'ensemble de la communauté scientifique va accroître ce manque de représentations spécifiques. En effet, ces ressources de calculs ouvrent le monde de la simulation numérique et de la visualisation scientifique *a fortiori* à de nouvelles branches de la recherche. Ces nouveaux domaines auront besoin de leurs propres techniques de visualisation qui reposeront sur l'expérience et les objectifs des utilisateurs. Il est donc primordial que le domaine soit à l'écoute des utilisateurs. De ces dialogues, la visualisation scientifique pourra alors élaborer de nouvelles techniques de visualisation spécialisées adaptées aux besoins.

Besoins des utilisateurs La visualisation scientifique doit ainsi répondre à des besoins spécifiés par des utilisateurs experts dans des domaines scientifiques précis mais variés. Il est donc important que le domaine sache évaluer ses nouvelles techniques de prétraitement, de stockage en mémoire et de rendu par rapport à ces besoins. L'évaluation des techniques développées par une communauté scientifique est en effet indispensable, d'autant plus quand celles-ci sont appliquées à des problèmes concrets qui ont besoin d'une solution répondant à des critères précis. Pour atteindre cet objectif, elle doit multiplier les partenariats avec le monde industriel comme la recherche publique (en physique, chimie, écologie, ...), principaux utilisateurs des techniques développées par la communauté de la visualisation scientifique.

En effet, de telles collaborations permettent, dans un premier temps, de comprendre les attentes des utilisateurs, grâce à la communication de leur problématique, de leurs habitudes et surtout de leurs *desirata* en terme de création d'images, d'interactivité et de manipulation de données. Il est alors possible d'établir de nouvelles techniques de visualisation et de les évaluer directement en terme d'efficacité par rapport à une problématique initiale donnée. La visualisation scientifique s'inscrit ainsi pleinement dans son rôle au sein de la boucle de la découverte scientifique.

Son évaluation repose donc sur sa capacité efficiente à produire des résultats exploitables pour les autres domaines. La création de tels partenariats assure ainsi un cercle vertueux. Les progrès de la visualisation scientifique assurent les progrès des domaines scientifiques utilisant la simulation numérique. Ceux-ci peuvent alors exiger à leur tour, grâce à l'expérience acquise, de nouvelles représentations ou des prétraitements originaux à la communauté de la visualisation scientifique.

Néanmoins, une des limitations actuelles pour répondre à ces besoins semble être la disponibilité et la diversité des ensembles de données libres de droit. La communauté mondiale de la visualisation scientifique ne travaille en effet que sur quelques dizaines d'ensembles de données depuis une vingtaine d'années. Cela est peu si on les compare aux centaines de maillages triangulaires disponibles pour la communauté graphique. De plus, certains ensembles de données sont fournis sans aucun détail sur les motivations ou les objectifs qui ont

[9] comme *RoadRunner* appartenant au Département de l'Énergie des États-Unis (DOE) situé à *Los Alamos, Nouveau Mexique*.

provoqué la réalisation d'une telle simulation. Il est alors difficile de concevoir des nouvelles approches sans finalité et de concevoir des images représentatives sans contexte.

Il est donc essentiel pour que la visualisation scientifique réponde pleinement aux besoins des utilisateurs que ceux-ci, en contrepartie, rendent disponibles des exemples de simulation de données concrets et détaillés. En effet, la restriction de la diversité est un frein à l'innovation et à la recherche car la communauté semble finalement ne répondre qu'aux problématiques liées uniquement aux ensembles de données actuellement disponibles (principalement médicaux). Une collaboration renforcée entre les différents laboratoires de visualisation (publics et industriels) mais aussi avec les laboratoires de calculs numériques (en plus des partenariats) pourrait favoriser les échanges de types de données et ainsi diversifier les domaines d'application, les représentations géométriques et les finalités.

Interdisciplinarité Enfin, la visualisation scientifique ne pourra relever certains de ses futurs défis sans investir (encore plus) dans l'interdisciplinarité sur laquelle elle s'appuie déjà fortement. L'algorithmique pour l'optimisation des structures de données et la réduction de la complexité, la géométrie algorithmique pour le prétraitement des maillages irréguliers et le graphique pour la production d'images, sont entre autres des domaines pleinement intégrés à la visualisation scientifique. Mais face aux révolutions architecturales des ordinateurs (dont les cartes graphiques), aux innovations technologiques et à l'expertise des utilisateurs, cela n'est plus suffisant.

L'un des buts de la visualisation scientifique est de produire des images porteuses de sens, c'est-à-dire qui répondent visuellement de manière précise aux objectifs de l'utilisateur en fonction du contexte de la simulation : découvrir des phénomènes locaux, comparer la simulation à des données réelles, garantir la fiabilité d'un objet dans une certaine situation, déterminer des zones d'incertitude... L'évaluation de l'impact de ces images sur la compréhension du phénomène est crucial. L'image est-elle ambiguë en terme de profondeur, y'a-t-il des occlusions gênantes, est-il possible de comprendre les interactions entre diverses variables, est-ce que la structure sous-jacente du modèle peut être reconstruite mentalement ? Cela revient à se poser la question plus générique suivante : la technique de visualisation est-elle adaptée à ma problématique ?

Une manière d'évaluer ces images et donc les techniques de visualisation – autres que *via* le retour direct des utilisateurs – est de déterminer leurs limitations perceptives afin de formuler des solutions plus adéquates. L'idéal serait de tendre vers des méthodes d'évaluation automatiques permettant de classifier l'efficacité des techniques par domaine avec pour finalité la réalisation d'une taxinomie. Cela ne sera possible que si la visualisation scientifique établit des liens forts avec le domaine de la perception, démarche en place depuis quelques années déjà [Bou09].

La création d'images ayant un sens peut aussi passer par le rapprochement de la visualisation scientifique avec le domaine du rendu non-photoréaliste (ou expressif) afin de créer des techniques de rendu se rapprochant des dessins techniques ou anatomiques, représentations déjà utilisées (et donc ancrées dans les habitudes) par un certain nombre de professionnels. Plusieurs solutions ont déjà été proposées dans cette direction principalement en médical et en mécanique *via* des extrusions ou des éclatements.

Un tel rapprochement ne peut qu'aider le domaine de la visualisation scientifique à proposer des techniques de rendu plus centrées sur l'extraction des zones d'intérêts. En effet, un rendu expressif (à l'inverse d'un rendu photoréaliste comme le rendu volumique direct) tente de simplifier la représentation en ne retenant que les caractéristiques probantes. Cela ne peut être réalisé qu'en définissant de manière précise des mesures de saillances, d'importance, de contenu d'informations au sein des ensembles de données. Ou plus fondamentalement à automatiser ce que l'être humain est capable de faire de manière naturelle : extraire, segmenter, localiser des informations au sein d'un environnement complexe.

De plus, ce domaine est un moteur d'innovation en terme de techniques de visualisation car il pose le problème même de l'association entre l'information et la représentation : comment représenter une surface, un champ vectoriel avec des techniques qui essayent d'être plus ou moins minimaliste (points, lignes, hachures, ombres, ...) ? Couplée avec une analyse perceptive, ce rapprochement ne peut que favoriser la création de nouvelles manières de visualiser des simulations.

La génération de gros ensembles de données, les grappes de cartes graphiques, les processeurs à l'architecture parallélisée, poussent aussi la visualisation scientifique à créer des rapprochements de plus en plus étroits

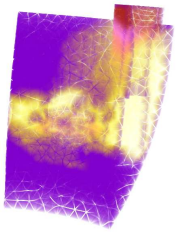
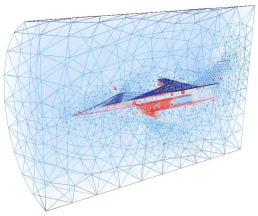
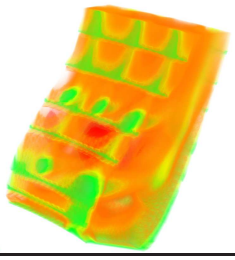
avec la communauté du parallélisme. Cette alliance, déjà existante dans de nombreux laboratoires français et étrangers, permet ainsi la création d'applications hautement parallèles et interactives afin de faciliter l'exploitation de données massives. La localisation des données au sein de serveurs externes (pouvant se trouver sur un autre continent) impose aussi des contraintes que des spécialistes du réseau ont sans doute en partie déjà résolues. Ces collaborations, déjà fructueuses depuis de nombreuses années, permettent ainsi de proposer des solutions efficaces pour la visualisation interactive distante de gros volumes de données en considérant la globalité du pipeline de visualisation du stockage des données jusqu'à leur visualisation. Elles permettront sans doute de créer et multiplier des solutions efficientes pour les problèmes de demain.

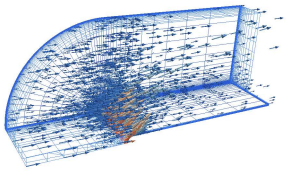
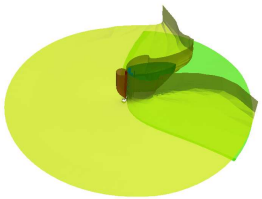
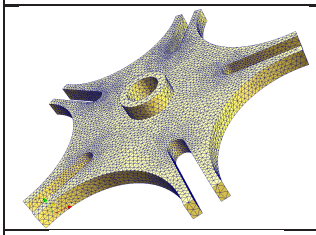
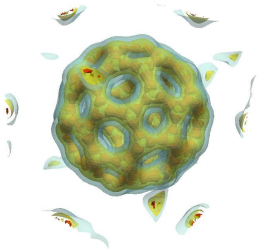
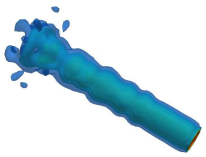
Enfin, la compréhension d'une simulation peut aussi passer par une représentation tridimensionnelle ou sensorielle au sein de laquelle un utilisateur pourrait interagir directement avec les résultats. Il pourrait ainsi injecter des particules pour suivre les directions d'un flux, toucher les isosurfaces afin d'en saisir les variations locales. Compléter la vision en utilisant d'autres sens (toucher, ouïe, ...) faciliterait sans doute la compréhension des phénomènes. L'idéal serait de trouver là encore des représentations intuitives pour des êtres humains ayant l'habitude d'analyser un monde environnant avec l'ensemble de leurs capacités sensorielles. Des collaborations avec les chercheurs en réalité virtuelle et augmentée existent déjà. Elles pourraient accroître rapidement le nombre de représentation permettant d'exploiter des résultats et ainsi faciliter la rapidité de compréhension des phénomènes dans les années à venir.

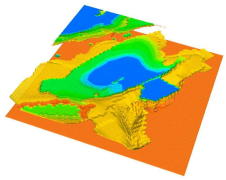
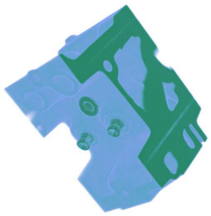
Ces exemples ne sont que des pistes de partenariats existants entre différents domaines qui ont fondamentalement beaucoup de choses à partager. Il est simplement essentiel que la visualisation scientifique collabore avec ses domaines applicatifs mais aussi avec des domaines annexes afin que la recherche puisse répondre aux attentes des utilisateurs. La visualisation scientifique pourra alors assurer pleinement le rôle de clef de voûte qui lui incombe de par sa position dans la boucle de la découverte scientifique.

ANNEXE A

Ensembles de Données

	**Nom** : *SPX* (Super Phénix) **Type** : Simulation **Origine** : *eDF*, courtoisie de Bruno Nitrosso **Taille** : 12 936 tétraèdres **Attributs** : Scalaire **Détails** : Simulation de la vitesse d'un liquide de refroidissement au sein du réacteur de la centrale *Super Phénix*. **URL** : http ://shapes.aim-at-shape.net/view.php ?id=629 **Note** : Existe aussi en version raffinée *SPX (r)* composée de 10 911 011 tétraèdres
	**Nom** : *Fighter* (Langley) **Type** : Simulation **Origine** : *Langley NASA*, courtoisie de R.W. Neely et J. T. Batina. **Taille** : 70 152 tétraèdres **Attributs** : Scalaire **Détails** : Grille tétraédrique entourant la moitié d'un avion de combat. Le champ est issu d'une simulation du déplacement de l'air autour de l'avion au sein d'une soufflerie. **URL** : http ://shapes.aim-at-shape.net/view.php ?id=632 **Note** : Existe aussi en version raffinée *Fighter (r)* composée de 5 929 085 tétraèdres
	**Nom** : *Comb* (Chambre de Combustion) **Type** : Simulation **Origine** : *VTK*, courtoisie de W. J. Schroeder **Taille** : 215 040 tétraèdres **Attributs** : Scalaire **Détails** : Simulation de la densité d'essence se trouvant au sein d'une chambre de combustion afin d'en vérifier le bon fonctionnement. **URL** : http ://www.vtk.org/VTK/resources/software.html

	**Nom** : *Blunt Fin* (Conduit d'aération) **Type** : Simulation **Origine** : *NASA* courtoisie de C.M. Hung et P.G. Buning **Taille** : 222 414 tétraèdres **Attributs** : Scalaire et Vectoriel **Détails** : Simulation du trajet d'un courant d'air enfermé au sein d'une conduite. Le flux est sensé être parallèle à la base. Les différents champs représente la norme et la vitesse de ce courant d'air. **URL** : http ://www.nas.nasa.gov/Research/Datasets/Hung/index.shtml
	**Nom** : *Post* (Liquid Oxygen Post) **Type** : Simulation **Origine** : *NASA* courtoisie de S. E. Rogers, D. Kwak et U. Kaul **Taille** : 616 050 tétraèdres **Attributs** : Scalaire **Détails** : Simulation de flux d'oxygène liquide incompressible au travers d'une plaque possédant un pilier cylindrique perpendiculaire à la plaque en son centre. Cette simulation est réalisée au sein d'un moteur de fusée afin de vérifier que la présence du pilier remplit son objectif : celui de mieux mélanger le flux. **URL** : http ://www.nas.nasa.gov/Research/Datasets/Rogers/index.shtml
	**Nom** : *Piece* (CAO Mécanique) **Type** : Maillage Tétraédrique **Origine** : *GMSH*, courtoisie de C. Greuzaine et J.F. Remacle **Taille** : 889 157 tétraèdres **Attributs** : Aucun **Détails** : Pièce mécanique, engrenage. **URL** : http ://geuz.org/gmsh/
	Nom : *Torso* **Type** : Acquisition **Origine** : *Université de l'Utah*, courtoisie de R. Kepfler **Taille** : 1 082 723 tétraèdres **Attributs** : Scalaire **Détails** : On étudie le champ bioélectrique produit par le cœur. Le champ scalaire représente ainsi des potentiels quasi constant au sein du torse excepté autour du coeur. **URL** : http ://software.sci.utah.edu/
	**Nom** : *BuckyBall* **Type** : Simulation **Origine** : *AVS International Center* **Taille** : 1 250 235 tétraèdres (issu d'une grille régulière) **Attributs** : Scalaire **Détails** : Structure de cage fermée des fullerènes (rappelant celle d'un *ballon de football*). La molécule C_{60} est ainsi composée de 60 carbones disposés aux sommets d'un polyèdre régulier de 0,7 nm de diamètre et dont les facettes sont 20 hexagones et 12 pentagones. Le champ scalaire représente ici la densité de ces atomes de carbone. **URL** : http ://shapes.aim-at-shape.net/view.php?id=169
	**Nom** : *Fuel* **Type** : Simulation **Origine** : *German Research Council* **Taille** : 1 250 235 tétraèdres (issu d'une grille régulière) **Attributs** : Scalaire **Détails** : Simulation de l'injection d'essence au sein d'une chambre à combustion, le champ scalaire représente la densité de l'essence (inversement proportionnelle à la présence d'air) **URL** : http ://www.gris.uni-tubingen.de/edu/areas/scivis/volren/datasets/datasets.html

	Nom : *DTI* **Type** : Acquisition **Origine** : CT SCAN, courtoisie de S. Roettger **Taille** : 4 596 765 tétraèdres (issu d'une grille régulière) **Attributs** : Scalaire **Détails** : Scanner d'un cerveau humain **URL** : http ://www9.cs.fau.de/Persons/Roettger/
	**Nom** : *Sf1* **Type** : Simulation **Origine** : *the Quake Project* courtoisie de Dave O'Hallaron **Taille** : 13 980 162 tétraèdres **Attributs** : Scalaire et Vectoriel **Détails** : Le maillage représente la vallée de San Fernando au Sud de la Californie. Les champs associés représentent à la fois la densité du sol et la vitesse des ondes de propagation. **URL** : http ://www.cs.cmu.edu/ quake/meshsuite.html
	**Nom** : *Engine* **Type** : Acquisition **Origine** : *General Electric* **Taille** : 41 943 040 tétraèdres (issu d'une grille régulière) **Attributs** : Scalaire **Détails** : CT scanner de deux cylindres d'un moteur. **URL** : http ://www.gris.uni-tubingen.de/edu/areas/scivis/volren/datasets/datasets.html **Note** : Existe aussi avec une version seuillée selon l'isosurface 70 comportant 5 152 961 tétraèdres.

ANNEXE B

Temps de Précalculs

Afin de mettre en place notre schéma birésolution permettant l'extraction d'une zone à haute précision au sein d'une boule d'intérêt contrôlée par l'utilisateur tout en conservant une résolution grossière sur son pourtour, plusieurs étapes de précalculs peuvent être mis en place :

- *optionnel* : la construction d'un maillage grossier via l'algorithme de simplification présenté dans le chapitre 2, section 2.2 ;
- **obligatoire** : la détermination d'un partitionnement des sommets fins en fonction des sommets grossiers comme détaillé dans le chapitre 2, section 2.6 ;
- *optionnel* : le plongement du maillage tétraédrique au sein d'une grille régulière afin d'obtenir une visualisation *out-of-core* pour l'exploration de gros maillages de données, décrit au chapitre 3, section 4.

Nous présentons ci-après les temps de précalculs de la globalité de ces étapes de précalculs **en secondes**. Si les étapes de précalculs sont réalisés dans leur globalité de manière consécutives (simplification+partitionnement), la surface de bord peut être extraite qu'une seule fois. Cela revient à retrancher le temps de son extraction du temps de partitionnement.

	Bucky	Fighter (r)	Spx (r)	Sf1	Engine
# tétraèdres	1 250 235	5 929 085	10 911 011	13 980 162	41 943 040
Extraction de la Surface de Bord	26	287	586	930	1 192
Simplification	35	306	625	1 020	1 507
Partitionnement	28	360	1 518	990	1 230
Découpage Spatial	14	178	402	143	418
Temps Total (étapes indépendantes)	77 $1min17s$	844 $14min4s$	2 545 $42min25s$	2 143 $35min43s$	4 347 $1h12min27s$
Temps Total (simplif.+part. consécutifs)	51 $51s$	557 $9min17s$	1 959 $32min39s$	1 213 $20min13s$	3 155 $52min35s$

TABLE DES MATIÈRES

Introduction **11**
Contexte et Motivations . . . 12
Contributions . . . 13
Organisation du document . . . 13

1 Visualiser des Maillages Tétraédriques : Motivations et Enjeux **15**
1 Motivations . . . 16
1.1 Discrétisation de l'espace . . . 16
1.2 Simulation . . . 18
1.3 Visualisation . . . 19
2 Maillages Tétraédriques : Notions et Définitions . . . 20
2.1 Complexe Simplicial . . . 20
2.2 Critère de qualité . . . 21
2.3 Structures de Données . . . 24
3 Maillages Tétraédriques et Visualisation . . . 24
3.1 Exploitation et Diffusion . . . 25
3.2 Temps Réel et Interactivité . . . 25
3.3 Les différentes techniques de visualisation . . . 26
4 Carte Graphique : Notions et Définitions . . . 29
4.1 Historique et Motivations . . . 30
4.2 Le Pipeline Graphique . . . 30
4.3 Unification Architecturale et Devenir . . . 32
5 Enjeux . . . 32
5.1 La taille des données . . . 32
5.2 La localisation géographique des données . . . 34
5.3 Localité de l'information . . . 35
6 Conclusion et Objectifs . . . 36

2 Bi-Résolution : Concepts et Précalculs **37**
1 Simplification et MultiRésolution . . . 38
1.1 Simplification . . . 39
1.2 MultiRésolution . . . 42
2 Bi-Résolution : les étapes de précalculs . . . 48
2.1 Objectifs . . . 48

2.2 Création d'un maillage grossier . . . 48
2.3 Partition d'un maillage . . . 51
2.4 Validité topologique de la partition . . . 53
2.5 Partition issue d'un processus de simplification . . . 55
2.6 Partition issue de critères géométriques . . . 57
3 Conclusion et Perspectives . . . 62

3 Bi-Résolution : Extraction Dynamique **65**
1 Schéma Birésolution : Problématique et Construction . . . 66
1.1 Extraction de zones d'intérêt . . . 66
1.2 Extraction d'un maillage birésolution . . . 69
2 Accélération de l'extraction sur le processeur central . . . 73
2.1 Extraction de la zone d'intérêt . . . 74
2.2 Mise-à-jour de la zone d'intérêt . . . 74
2.3 Extraction du maillage birésolution . . . 76
2.4 Complexité et Résultats . . . 76
3 Implantation sur la carte graphique . . . 79
3.1 Mise en œuvre . . . 79
3.2 Résultats et Limitations . . . 80
4 Implantation en mémoire externe . . . 82
4.1 Principe des méthodes en mémoire externe . . . 82
4.2 Implantation . . . 83
4.3 Résultats et Discussions . . . 85
5 Bilan et Perspectives . . . 87

4 Visualiser via le Rendu Volumique **89**
1 État de l'Art . . . 90
1.1 Intégrale du rendu volumique . . . 90
1.2 Lancer de Rayons . . . 93
1.3 Méthodes projectives . . . 94
1.4 Méthodes Hybrides . . . 97
1.5 Comparaisons et Discussions . . . 97
2 Tri des primitives et cohérence temporelle . . . 99
2.1 Objectifs . . . 99
2.2 SXMPVO Cohérent . . . 99
2.3 *Projected Tetrahedra* au sein d'un processeur de primitives . . . 102
2.4 Résultats et Discussion . . . 103
3 Méthode Projective sur Carte Graphique . . . 105
3.1 Problématique . . . 105
3.2 Rendu Volumique Direct . . . 106
3.3 Résultats et Discussion . . . 108
4 Autres Techniques de Visualisation . . . 113
4.1 Intégration des Isosurfaces . . . 114
4.2 Résultats et Discussion . . . 115
5 Bilan et Perspectives . . . 116

Conclusion **119**
Résumé des contributions . . . 119
Perspectives . . . 120

A Ensembles de Données **125**

B Temps de Précalculs **129**

Table des matières **131**

Table des figures **135**

Index **138**

Bibliographie **141**

TABLE DES FIGURES

1.1 Boucle de la découverte scientifique . . . 17
1.2 Cellules pouvant composée un maillage volumique . . . 18
1.3 Maillage hybride pour la simulation de fluides . . . 18
1.4 Maillage hexaédrique pour la simulation de gaz . . . 18
1.5 Exemple des étapes de création d'un maillage tétraédrique pour une pièce mécanique . . . 19
1.6 Simplexes dans $\mathbb{R}^3$ . . . 20
1.7 Lien et Étoilé d'un sommet dans un maillage triangulé . . . 20
1.8 Nerf d'un ensemble quelconque . . . 21
1.9 Basculement d'arêtes . . . 23
1.10 Tétraèdre contraint de Delaunay . . . 23
1.11 Marching Tetrahedra . . . 26
1.12 Techniques de visualisation pour un champ scalaire . . . 28
1.13 Techniques de visualisation pour un champ vectoriel . . . 28
1.14 Méthodes denses texturées (LIC) . . . 29
1.15 Pipeline graphique . . . 31
1.16 Pipeline programmable des cartes graphiques . . . 31
1.17 Stratégies de tri pour la visualisation scientifique sur grappe . . . 34
1.18 Exemples de la localité de zones d'intérêt compactes . . . 35

2.1 Simplification surfacique de la peau d'une pièce mécanique . . . 38
2.2 Décimation d'arêtes : l'opération *half-edge collapse* . . . 39
2.3 Décimation de tétraèdres : l'opération *tetfuse* . . . 39
2.4 Simplification volumique d'un maillage tétraédrique . . . 41
2.5 Multirésolution volumique d'un maillage tétraédrique . . . 43
2.6 Erreurs d'approximation dynamique pour l'extraction de niveau de détails . . . 44
2.7 Graphe Direct Acyclique pour la multirésolution . . . 45
2.8 Hiérarchie de segments sur un maillage tétraédrique . . . 46
2.9 Limitations de la multirésolution basées sur les opérations locales . . . 47
2.10 Étapes de la simplification des maillages tétraédriques . . . 49
2.11 Conservation de la topologie du champ scalaire . . . 51
2.12 Maillage Birésolution . . . 52
2.13 *K-way Partitioning* de maillages tétraédriques . . . 53
2.14 Premier exemple de partition n'assurant pas la reconstruction du maillage grossier . . . 54
2.15 Second exemple de partition n'assurant pas la reconstruction du maillage grossier . . . 54
2.16 Partition issue d'un processus de simplification . . . 55

2.17 Exemples de partitions issues d'un processus de simplification 55
2.18 Limitation de l'approche reposant sur un algorithme de simplification 56
2.19 Validation géométriquement faible 57
2.20 Comparaisons entre le diagramme de Voronoï et les coordonnées barycentriques 58
2.21 Problèmes pouvant être responsables de la non conformité de conditions pour les coordonnées barycentriques 60
2.22 Exemple de condition non vérifiée pour la partition basée sur les coordonnées barycentriques . 61
2.23 Intersection entre les tétraèdres de liens et les tétraèdres fins 61

3.1 Étapes de segmentation de données médicales régulières 67
3.2 Comparaison entre *MagicSphere* et *BiRes* 67
3.3 Transformation des cellules de lien pour un maillage triangulaire 69
3.4 Intersection vide entre maillage fin et grossier 71
3.5 Localité de l'extraction 72
3.6 Octree construit sur un maillage tétraédrique 73
3.7 Extraction d'un maillage birésolution tétraédrique 73
3.8 Tétraèdres actifs au sein de la zone d'intérêt 75
3.9 Mise à jour de la zone d'intérêt en utilisant la cohérence temporelle 75
3.10 Complexité de la mise-à-jour de la zone d'extraction fine 77
3.11 Exemple d'exploration au sein de l'ensemble de données *BuckyBall* 78
3.12 Extraction du maillage birésolution sur carte graphique 79
3.13 Temps d'extraction du maillage birésolution sur la carte graphique pour *Fighter (r)* 81
3.14 Pagination utilisée pour notre implantation en mémoire externe 83
3.15 Temps d'extraction pour la méthode en mémoire externe 86
3.16 Maillage birésolution pour l'ensemble de données *Engine* 88

4.1 Intégrale du rendu volumique dans le cadre du modèle émission-absorption 91
4.2 Pré- et Post-Classification 92
4.3 Lancer de Rayons 93
4.4 Ordre de visibilité 95
4.5 Cycle de Visibilité 95
4.6 Projected Tetrahedra 97
4.7 Paradigme de Balayage 97
4.8 Comparaisons des techniques de rendu volumique direct ordonné 98
4.9 Mise-à-jour des relations de visibilité au sein du maillage birésolution 100
4.10 Détermination des relations d'adjacence entre tétraèdres de lien 100
4.11 Correspondance entre A-Buffer et Depth-Peeling 101
4.12 Implantation de Projected Tetrahedra au sein d'un processeur de primitives 101
4.13 Exemples de rendu volumique direct avec SXMPVO accéléré et PT au sein d'un processeur de primitives 102
4.14 Comparaisons entre SXMPVO avec et sans cohérence 104
4.15 Comparaison entre HAVS pour un maillage statique et dynamique 106
4.16 Tri et méthode projective sur carte graphique 107
4.17 Exemples de rendu volumique direct avec notre méthode sur carte graphique 109
4.18 Évaluation de l'erreur d'approximation pour un nombre limité de couches 110
4.19 Évaluation de l'erreur d'approximation suite à un rejet de primitives 111
4.20 Comparaison de notre méthode avec HAVS 112
4.21 Intégration du rendu volumique direct au sein du schéma birésolution 113
4.22 Techniques de visualisation usuelles intégrées au sein du schéma birésolution 114
4.23 Intégration des isosurfaces au sein du schéma birésolution 116

LISTE DES TABLEAUX

1.1 Tétraèdres de différentes qualités 22
1.2 Temps-réel et interactivité 25
1.3 Exemples de simulations massives 33

2.1 Temps de simplification de maillages tétraédriques des précédentes approches 42
2.2 Occupation mémoire et temps de précalculs pour les structures implicites 46
2.3 Flux d'extraction des primitives pour la mise à jour des niveaux de détails 47
2.4 Temps de simplification de maillages tétraédriques 50
2.5 Mesures d'erreurs et mémoire de la simplification de maillages tétraédriques 51
2.6 Temps de calcul pour extraire la partition du processus de simplification 56
2.7 Temps d'extraction de la partition basée sur les coordonnées barycentriques 59

3.1 Temps d'extraction de la zone d'intérêt avec cohérence temporelle 78
3.2 Temps de précalculs pour la construction de la structure en mémoire externe 85

4.1 Temps d'affichage des techniques de rendu volumique direct ordonné 98
4.2 Comparaisons de SXMPVO original et architecturalement accéléré 103
4.3 Comparaisons temporelles entre *PTINT* et notre implantation 103
4.4 Temps du rendu en fonction d'un nombre limité de couches 110
4.5 Évaluation quantitative et temporelle du rejet de primitives 111

INDEX

A

Algorithme
- Focus+Contexte 36, 66
- Mémoire Externe 24, 33
- MultiRésolution 36, 42
- Parallèle 33
- Simplification 39

AMR voir Structure de Données
API voir Interface de Programmation

B

Boule d'intérêt voir Zone d'intérêt
BSP-Tree 82

C

CAO 17, 19, 23
Carte Graphique 29, 79
- Attribut 30
- Buffer 31
- Fragment 30, 33
- Geometry Shader 31
- Primitive 30, 33
- Shader 32
- Texels 31
- Texture 31
- Unifiée 30
- Vertex Shader 31

Cluster 53
- Actif 70

Clusterisation voir Partition
Cohérence Temporelle 75, 100, 104
Complexité 76
Composition
- Back-To-Front 92

Coordonnées Barycentriques 58
Critère de Delaunay voir Maillage
Croissance de Regions 74, 75

D

Définition
- Cellule 16
- Maillage 16

DAG voir Structure de Données
Décimation
- Arêtes 39
- Sommet 49
- Tétraèdres 40

Définition
- Arête 20
- Cellule 20
- Complexe Simplicial 20
- Étoilé 20
- Face 20
- Isosurface 27
- Lien 20
- Nerf 21
- Point 20
- Points Critiques 49
- Simplexe 20
- Tétraèdre 20
- Triangle 20
- Visibilité 23

Depth-Peeling 94, 95, 101, 106
Diagramme de Voronoï 23, 58
Données
- Grilles Irrégulières 66
- Grilles Régulières 68

E

Effondrement d'Arêtes 39, 55
Éléments Finis 39
Ensemble de Données

BuckyBall 44, 51, 102, 114
Comb 28, 49, 53, 109
DTI 114
Fuel 114
Pièce 19, 38, 53
SPX 43, 47, 73, 102
Sf1 109, 114
Blunt Fin 28, 35, 67
BuckyBall 78
Comb 28
Engine 88
Torso 35
Erreur d'Approximation 40
Attributs 41
Géométrique 40
Qualité 41
Topologique 40
Erreur Dynamique 43
Étoilé 20, 56

F

FLOPS 30
Focus+Contexte voir Algorithme
Fonction de Transfert 26, 90
Définition 91

G

GPGPU 32
GPU voir Carte Graphique
Grappe 32, 33
Grappes d'Ordinateurs 37
Grille Régulière 17

I

Intégrale du Rendu Volumique
Continue 90
Discrétisée 91
Interactivité voir Taux de Rafraîchissement
Interface de Programmation 80, 82
DirectX 30
OpenGL 30
Interval Tree 33

K

Kd-Tree 48

L

LIC 28
Lien 20, 21, 40
Ligne de Courant
Pathlines 28
Streaklines 28
Streamlines 28, 33
Streamribbons 28
LOD voir Niveaux de Détails

M

Maillage
Conforme 44, 45, 52
Critère de Delaunay 22
Delaunay 48
Flip 22
Qualite 21
Relations d'adjacences 73
Maillage Hybride 17
Marching Cube 27
Marching Tetrahedra 27
Mémoire Externe 33, 82
Bloc 82
Cache 83, 84
Déréférencement 82
Prefetch 83
Structures de Données 82
MIP 27
Multi-Triangulations 45
MultiRésolution 42
Front actif 44
Mise à jour 44

N

Nerf 21, 55
Niveaux de Détails 42

O

Octree 33, 60, 74–76, 82
Ordre de visibilité 94
Out-of-Core voir Mémoire Externe
Outils
Paraview 97
Engrid 17
GMSH 17
HAVS 96
HRC 98
Metro 50
Tetgen 50
TetMesh Comparator 50

P

Parallélisation 37
Parallèlisme
Sort-First 33

Sort-Last 33
Partition
Cluster 53
Définition 53
Problème d'optimisation 53
Pipeline Graphique 30
Projected Tetrahedra 96, 102

Q

QEM voir Erreur d'Approximation

R

R-tree 33
Rayons X 27
Region Growing 74, 75
Rendu Volumique
Composition 92
Intégrale Continue 90
Intégrale Discrétisée 91
Intégrale Discrète 92
Interpolation 92
Lancer de Rayons 68, 93
Modèle Émission-Absorption 90
Splatting 96

S

Segmentation 68
Shader
Geometry 79, 103, 106
Simplexe voir Définition
de Delaunay 22
Simplexes 20, 39
Simplification 39
Simulation 18, 33
Structure de Données
AMR 36
Directed Acyclic Graph 44, 45
Explicite 45
Forêt d'Arbres Binaires 44
Hiérarchie de Segments 45
Implicite 45
Soupe de Cellules 24, 80
Structure Indexée 24
Structure Topologique 24
Supercalculateur 32

T

T-BON 33
Tétraèdre
Actif 70
De Lien 70
Dégénéré 70
Inactif 70
Taux de Rafraîchissement
Interactivité 25
Quasi-Temps-Réel 25
Temps-Réel 25
Techniques
Décimation voir Décimation
Visualisation voir Visualisation
Tetfuse voir Décimation de tétraèdres
Tetraédralisation
Contrainte de Delaunay 24
de Delaunay 22
Tetraèdre
Cap 22
Needle 22
Sliver 21, 22, 40, 41, 59
Wedge 22
Angle Diédral 22
Angle Solide 22
Contraint de Delaunay 23
Ratio d'Aspect 22
Ratio Rayon-Arête 22

V

Validité
Géométrique 57
Topologique 55
Visualisation 19, 25
Fonction de Transfert 26, 35
Hedgehog 27
Isosurface 27, 114
Ligne de Courant 28
Line Integral Convolution 28
Plan de Coupe 27, 113
Rendu Volumique Direct 27
Voronoï voir Diagramme de Voronoï
Voxel 17, 83

Z

Zone d'intérêt 35, 66
Complexité 76
Croissance de Regions 74, 75
Définition 70
Source 74

BIBLIOGRAPHIE

[ABM+01] James Ahrens, Kristi Brislawn, Ken Martin, Berk Geveci, C. Charles Law, and Michael Papka. Large-scale data visualization using parallel data streaming. *IEEE Comput. Graph. Appl.*, 21(4) :34–41, 2001. 37

[ACS+07] Erik W. Anderson, Steven P. Callahan, Carlos E. Scheidegger, John M. Schreiner, and Claudio T. Silva. Hardware-assisted point-based volume rendering of tetrahedral meshes. In *Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing*, pages 163–172, 2007. 96, 98, 105

[Aur91] Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3) :345–405, 1991. 58

[AYM99] Ph Ackerer, A. Younes, and R. Mose. Modeling variable density flow and solute transport in porous medium : 1. numerical model and verification. *Transport in Porous Media*, 35(3) :345–373, 1999. 19

[BCL06] Luc Buatois, Guillaume Caumon, and Bruno Lévy. Gpu accelerated isosurface extraction on tetrahedral grids. In *Proceedings of the International Symposium on Visual Computing (ISVC)*, volume 4291, pages 383–392, 2006. 27, 114

[BE95] M Bern and D. Eppstein. Mesh generation and optimal triangulation. *Lecture Notes Series on Computing, Computing in Euclidian Geometry -2nd edition*, 4 :47–123, 1995. 21

[BKKW08] Kai Bürger, Polina Kondratieva, Jens Krüger, and Rüdiger Westermann. Importance-driven particle techniques for flow visualization. In *Proceedings of IEEE VGTC Pacific Visualization Symposium 2008*, 2008. 68

[BKS97] Paul Bunyk, Arie Kaufman, and Claudio T. Silva. Simple, fast, and robust ray casting of irregular grids. In *Proceeding of the Dagstuhl, Scientific Visualization Conference*, pages 30–36, 1997. 94, 97, 98

[BM08] M. Bavoil and K. Myers. Order independant transparency with dual depth peeling. Technical report, NVIDIA Corporation, 2008. 95, 113

[Bou09] Christian Boucheny. *Visualisation scientifique interactive de grands volumes de données : pour une approche perceptive*. PhD thesis, Université Joseph Fourier, 2009. 17, 66, 122

[BPCS06] Fabio F. Bernardon, Christian A. Pagot, Joao L. D. Comba, and Claudio T. Silva. Gpu-based tiled ray casting using depth peeling. *Journal of Graphics Tools*, 11(3) :23–29, 2006. 94, 98, 105

[BSP+93] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses : the see-through interface. In *SIGGRAPH '93 : Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 73–80, 1993. 67

[CA92] A. O. Cifuentes and Kalbag A. A performance study of tetrahedral and hexahedral elements in 3-d finite element structure. *Finite Element Analysis Design*, 12(3–4) :313–318, 1992. 17

[Car84] Loren Carpenter. The a-buffer, an antialiased hidden surface method. *SIGGRAPH Comput. Graph.*, 18(3) :103–108, 1984. 95, 99

[Cas06] Laurent Castanié. *Visualisation de Données Volumiques Massives – Application aux données sismiques*. PhD thesis, Université de Nancy, 2006. 34, 37

[CB04] Marcelo Cohen and Ken Brodlie. Focus and context for volume visualization. In *TPCG '04 : Proceedings of the Theory and Practice of Computer Graphics 2004 (TPCG'04)*, pages 32–39, 2004. 67

[CCM+00] Paolo Cignoni, D. Constanza, C. Montani, C. Rocchini, and R. Scopigno. Simplification of tetrahedral meshes with accurate error evaluation. In *Proceedings of VIS '00*, pages 85–92, 2000. 34, 40, 41, 42

[CCS05] Steven P. Callahan, João L. D. Comba, and Claudio T. Silva. Interactive rendering of large unstructured grids using dynamic level-of-detail. In *Proceedings of IEEE Visualization'05*, 2005. 98, 105

[CDE+00] S. W. Cheng, T. K. Dey, H Edelsbrunner, M. A. Facello, and S.-H. Teng. Sliver exudation. *Journal of ACM*, 47(5) :883–904, 2000. 21

[CDF98] Paolo Cignoni and Leila De Floriani. Power diagram depth sorting. In *Proceedings of the 10th Canadian Conference on Computational Geometry*, pages 88–96, 1998. 95

[CDFM+94] Paolo Cignoni, Leila De Floriani, Claudio Montani, Enrico Puppo, and Roberto Scopigno. Multiresolution modeling and visualization of volume data based on simplicial complexes. In *Proceedings of VVS '94*, pages 19–26, 1994. 67, 68, 115

[CDFM+04] Paolo Cignoni, Leila De Floriani, Paola Magillo, Enrico Puppo, and Roberto Scopigno. Selective refinement queries for volume visualization of unstructured tetrahedral meshes. *IEEE Transactions on Visualization and Computer Graphics*, 10(1) :29–45, 2004. 34, 36, 43, 45, 46, 47, 72, 73, 77

[CDFMP00] Paolo Cignoni, Leila De Floriani, Paola Magillo, and Enrico Puppo. Tan2 - visualization of large irregular volume datasets. Technical report, DISI, Departement of Computer and Information Science, University of Genova, 2000. 44

[CDM04] B. Cutler, J. Dorsey, and L. McMillan. Simplification and improvement of tetrahedral models for simulation. In *Proceedings of SGP '04*, pages 93–102, 2004. 41, 42, 59

[CE97] M. B. Cox and D. Ellsworth. Application-controlled demand paging for out-of-core visualisation. In *Proceedings of IEEE Visualization 97*, pages 235–244, 1997. 33

[CGG+04] Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio, and Roberto Scopigno. Adaptive tetrapuzzles : efficient out-of-core construction and visualization of gigantic multiresolution polygonal models. *ACM Trans. Graph.*, 23(3) :796–803, 2004. 82

[CGM+06] Andrej Cedilnik, Berk Geveci, Kenneth Moreland, James Ahrens, and Jean Favre. Remote large data visualization in the paraview framework. In L. P. Santos A. Heirich, B. Raffin, editor, *Eurographics Parallel Graphics and Visualization*, pages 162–170, 2006. 37

[CICS05] Steven P. Callahan, M. Ikits, João L. D. Comba, and C. Silva. Hardware-assisted visibility ordering for unstructured volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 11(3) :285–295, 2005. 96, 97, 98, 105, 110

[CKM+99] João L. D. Comba, James T. i Klosowsk, Nelson Max, Joseph S. B. Mitchell, Claudio T. Silva, and Peter L. Williams. Fast polyhedral cell sorting for interactive rendering of unstructuredgrids. In P. Brunet and R. Scopigno, editors, *Computer Graphics Forum (Eurographics '99)*, volume 18(3), pages 369–376. The Eurographics Association and Blackwell Publishers, 1999. 95

[CL03] Y. Chiang and X. Lu. Progressive simplification of tetrahedral meshes preserving all isosurface topologies. *Computer Graphics Forum*, 22 :493–504., 2003. 50

[Cla76] J. H. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10) :547–554, 1976. 33

[CM02] Prashant Chopra and Joerg Meyer. Tetfusion : An algorithm for rapid tetrahedral mesh simplification. *Proceedings of VIS '02*, 0 :133–140, 2002. 40, 42

[CM03] Prashant Chopra and Joerg Meyer. Topology sensitive volume mesh simplification with planar quadric error metrics. In *IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP 2003)*, pages 908–913, 2003. 40

[CMRS03] Paolo Cignoni, Claudio Montani, Claudio Rocchini, and Roberto Scopigno. External memory management and simplification of huge meshes. *IEEE Transactions on Visualization and Computer Graphics*, 9(4) :525–537, 2003. 33, 82

[CMSW04] Richard Cook, Nelson Max, Claudio T. Silva, and Peter L. Williams. Image-space visibility ordering for cell projection volume rendering of unstructured data. *IEEE Transactions on Visualization and Computer Graphics*, 10(6) :695–707, 2004. 95, 99

[CPHM04] Keith K. H. Choy, John F. Porter, Chi-Wai Hui, and Gordon McKay. Process design and feasibility study for small scale msw gasification. *Chemical Engineering Journal*, 105(1-2) :31–41, 2004. 18

[CS97] Y.-J. Chiang and C. T. Silva. I/o optimal isosurface extraction. In *Proceedings of IEEE Visualization'97*, pages 293–300, 1997. 82

[CSS98] Y.-J. Chiang, C. T. Silva, and W. J. Schroeder. Interactive out-of-core isosurface extraction. In *IEEE Visualization'98*, pages 167–174, 1998. 33, 82, 86

[DCH88] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. In *SIGGRAPH '88 : Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 65–74, 1988. 27, 67

[DDF02] E. Danovaro and L. De Floriani. Half-edge multi tessellation : a compact representation for multiresolution tetrahedral meshes. In *Proceedings 1st International Symposium on 3D Data Processing Visualization and Transmission, IEEE Computer Society*, pages 494–499, 2002. 45, 46

[DDFMP01] E. Danovaro, L. De Floriani, P. Magillo, and E. Puppo. *Digital and Image Geometry, Lectures Notes in Computer Science*, volume 2243, chapter Representing Vertex-Based Simplicial Multi-Complexes, pages 127–138. Springer Berlin, 2001. 45, 46

[DDFMP02] E. Danovaro, L. De Floriani, P. Magillo, and E. Puppo. Data structures for 3d multi-tesselations : an overview. In G.P. Bonneau In F.H. post and G.M. Nielson, editors, *Dagstuhl Scientific Visuqlizqtion Seminar*, pages 239–256. Kluwer Academic Publisher, 2002. 44, 45

[DEGN99] Tamal K. Dey, Herbert Edelsbrunner, Sumanta Guha, and Dmitry V. Nekhayev. Topology preserving edge contraction. *Publ. Inst. Math.*, 66 :23–45, 1999. 40

[DFH05] Leila De Floriani and Annie Hui. Data structures for simplicial complexes : an analysis and a comparison. In *Third Eurographics Symposium og Geometry Processing (SGP)*, pages 119–128, 2005. 24

[DFMP00] Leila De Floriani, Paola Magillo, and Enrico Puppo. The mt (multi-tesselation) package. Technical report, DISI, Departement of Computer and Information Science, University of Genova, http ://www.disi.unige.it/person/MagilloP/MT, 2000. 43, 44

[DGH03] Helmut Doleisch, Martin Gasser, and Helwig Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Proceedings of VISSYM '03*, pages 239–248, 2003. 68

[DH93] Thierry Delmarcelle and Lambertus Hesselink. Visualizing second-order tensor field with hyperstreamlines. *IEEE Computer Graphics Applications*, 13(4) :25–33, 1993. 29

[DP02] Christopher DeCoro and Renato Pajarola. Xfastmesh : fast view-dependent meshing from external memory. In *VIS '02 : Proceedings of the conference on Visualization '02*, pages 363–370, 2002. 33

[Ede01] Herbert Edelsbrunner. *Geometry and Topology of Grid Generation*. Cambridge University Press, Spring 2001. 21, 56

[EKE01] K. Engel, M. Kraus, and T. Ertl. High-quality pre-intergrated volume rendering using hardware-accelerated pixel shading. In *Proceedings of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pages 9–16, 2001. 93

[Eve01] C. Everitt. Interactive order-independent transparency. Technical report, NVIDIA Corporation, may 2001. 95, 101

[FG98] Anton Fuhrmann and Eduard Gröller. Real-time techniques for 3d flow visualization. In *Proceedings of the conference on Visualization '98*, pages 305 – 312, 1998. 68

[FK03] Randima Fernando and Mark. J. Kilgard. *The Cg Tutorial : The Definitive Guide to Programmable Real-Time Graphics*. Addisson-Wesley Educational Publisher Inc., 2003. 32

[FMS00] Ricardo Farias, Joseph S. B. Mitchell, and Cláudio T. Silva. Zsweep : an efficient and exact projection algorithm for unstructured volume rendering. In *VVS '00 : Proceedings of the 2000 IEEE symposium on Volume visualization*, pages 91–99, 2000. 97, 98

[FMSW00] Ricardo Farias, Joseph S.B. Mitchell, Claudio T. Silva, and Brian Wylie. Time-critical rendering of irregular grids. In *Proceedings of XIII Brazilian Symposium of Computer and Image Processing*, pages 243–250, 2000. 98, 105

[FVDF96] J. D. Foley, A. Van Dam, and S. K. Feiner. *Computer Graphics – Principles and Practice*. Addison-Wesley, 1996. 90

[Gar90] M. P. Garrity. Raytracing irregular volume data. *Computer Graphics (San Diego Workshop on Volume Visualization)*, 24(5) :35–40, 1990. 94

[Gas04] Martin Gasser. Fast focus+context visualization of large scientific data. In *CESCG 2004 : 8th Central European Seminar on Computer Graphics*, 2004. 68

[GGS99] Stefan Gumhold, Stefan Guthe, and Wolfgang Straßer. Tetrahedral mesh compression with the cut-border machine. In *VIS '99 : Proceedings of the conference on Visualization '99*, pages 51–58, 1999. 34, 46

[GH95] A. Guéziec and R. Hummel. Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 1(4) :328–342, 1995. 27, 114

[GH97] M. Garland and P. S. Heckbert. Surface simplification using quadrics error metrics. In *Proceedings of th 24th Annual Conference on Computer Graphics, SIGGRAPH'97*, pages 209–216, 1997. 40, 49

[GHLM05] Naga K. Govindaraju, Michael Henson, Ming C. Lin, and Dinesh Manocha. Interactive visibility ordering and transparency computations among geometric primitives in complex environments. In *Proceedings of SI3D '05*, pages 49–56, 2005. 96, 113

[Gie92] Christopher Giertsen. Volume visualization of sparse irregular meshes. *IEEE Computer Graphics and Applications*, 12(2) :40–48, 1992. 97

[GM08] Enrico Gobbetti and Fabio Marton. Far voxels : a multiresolution framework for interactive rendering of huge complex 3d models on commodity graphics platforms. In *SIGGRAPH Asia '08 : ACM SIGGRAPH ASIA 2008*, pages 1–8, 2008. 82

[GR86] Vivette Girault and Pierre-Arnaud Raviart. *Finite Element Methods For Navier-Stokes Equations : Theory and Algorithms*, volume 5. Springer-Verlag, 1986. 18

[GR09] C. Geuzaine and J.-F. Remacle. Gmsh : a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, Accepted for publication, 2009. 17

[Hau05] H. Hauser. Generalizing focus+context visualization. In *Scientific Visualization : The Visual Extraction of Knowledge From Data*, Mathematics+Visualization, pages 305 – 328. Springer, 2005. 67

[HB84] C. M. Hung and P. G. Buning. Simulation of blunt-fin induced shock wave and turbulent boundary layer separation. In *AIAA Aerospace Sciences Conference*, number Paper 84-0457, page Juanary, 1984. 28

[HDD+93] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *20th annual conference on Computer graphics and interactive techniques*, pages 19 – 26, 1993. 39

[HG97] P. S. Heckbert and M. Garland. Survay of polygonal surface simplification algorithms. Technical report, Carnegie Mellon University, 1997. 39

[HH92] Charles D. Hansen and Paul Hinker. Massively parallel isosurface extraction. In *VIS '92 : Proceedings of the 3rd conference on Visualization '92*, pages 77–83, 1992. 34

[HKRS+06] Markus Hadwiger, Joe M. Kniss, Christof Rezk-Salama, Daniel Weiskopf, and Klaus Engel. *Real-Time Volume graphics*. A. K. Peters, Ltd., 2006. 93

[HLK01] Jeffrey Ho, Kuang Chih Lee, and David Kriegman. Compressing large polygonal models. In *VIS '01 : Proceedings of the conference on Visualization '01*, pages 357–362, 2001. 33

[HMS95] Wolfgang Heidrich, Michael McCool, and John Stevens. Interactive maximum projection volume rendering. In *VIS '95 : Proceedings of the 6th conference on Visualization '95*, page 11, 1995. 27

[HS89] William Hibbard and David Santek. Interactivity is the key. In *VVS '89 : Proceedings of the 1989 Chapel Hill Worshop on Volume Visualization*, pages 39–43, 1989. 65, 66

[IDBL06] Franck P. Incropera, David P. DeWitt, Theodore L. Bergman, and Adrienne S. Lavine. *Fundamentals of heat and mass transfer*. Number 6. John Wiley, 2006. 18

[IG03] Martin Isenburg and Stefan Gumhold. Out-of-core compression for gigantic polygon meshes. *ACM Trans. Graph.*, 22(3) :935–942, 2003. 33

[IL05] Martin Isenburg and Peter Lindstrom. Streaming meshes. In *Proceedings of Visualization'05*, pages 231–238, 2005. 83

[ILGS06] Martin Isenburg, Peter Lindstrom, Stefan Gumhold, and Jonathan Shewchuk. Streaming compression of tetrahedral volume meshes. In *GI '06 : Proceedings of the 2006 conference on Graphics interface*, pages 115–121, 2006. 34

[JLZ+01] C. R. Johnson, Y. Livnat, L. Zhukov, D. Hart, and G. Kindlmann. Computational field visualization. In B. Engquist and W. Schmid, editors, *Mathematics Unlimited – 2001 and beyond*, volume 2, pages 605–630. Springer-Verlag, 2001. 29

[Joe91] B. Joe. Construction of three-dimensional delaunay triangulations from local transformations. *Computer Aided Geometric Design*, 8 :123–142, 1991. 23

[KD98] G. Kindlmann and J. W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *9th IEEE Visualization Conference (Vis'98)*, pages 79–86, 1998. 35

[KE01] Martin Kraus and Thomas Ertl. Cell-projection of cyclic meshes. In *VIS '01 : Proceedings of the conference on Visualization '01*, pages 215–222, 2001. 95

[Kes09] John Kessenich. *The OpenGL Shading Language, version 1.4*, Feb 2009. 32

[KHO+85] H. W. Kroto, J. R. Heath, S. C. O'Brien, R. F. Curl, and R. E. Smalley. C60 : Buckminsterfullerene. *Nature*, 318 (No. 6042) :162–163, 1985. 78

[Kin04] G. Kindlmann. Superquadric tensor glyphs. In *Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization 2004*, pages 147–154, 2004. 29

[KK98] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. In *Journal of Parallel and Distributed Computing 48*, pages 98–129, 1998. 53

[KKM96] Y. Kallinderis, A. Khawaja, and H. McMorris. Hybrid prismatic/tetrahedral grid generation for complex geometry. *AIAA Computational Fluid Dynamics Conference*, 34(2) :291–298, 1996. 17

[kowG09] krhonos openCL working Group. *The OpenCL Specification*. version 1.0, editor : aaftab munshi edition, 2009. 32

[KSH03] Ralf Kahler, Mark Simon, and Hans-Christian Hege. Interactive volume rendering of large sparse data sets using adaptive mesh refinement hierarchies. *IEEE Transactions on Visualization and Computer Graphics*, 09(3) :341–351, 2003. 35, 36

[KSS08] Roy Koomullil, Bharat Soni, and Rajkeshar Singh. A comprehensive generalized mesh system for cfd applications. *Mathematics and Computers in Simulations*, 78(5-6) :605–617, 2008. 17

[KSW01] S. Krishnan, C. Silva, and B. Wei. A Hardware-Assisted Visibility-Ordering algorithm with applications to volume rendering. In *Data Visualization*, pages 233–242, 2001. 96

[KV06] Youngmin Kim and Amitabh Varshney. Saliency-guided enhancement for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5) :925–932, 2006. 68

[KVH84] J.T. Kajiya and B.P. Von Herzen. Ray tracing volume densities. In *Proceedings of ACM SIGGRAPH Conference*, pages 165–174, 1984. 27

[KW99] Gordon Kindlmann and David Weinstein. Hue-balls and lit-tensors for direct volume rendering of diffusion tensor fields. In *VIS'99 : Proceedings of the conference on Visualization 1999*, pages 183–189, 1999. 29

[LAK+98] D. Laidlaw, E. Ahrens, D. Kremers, M. Avalos, R. Jacobs, and C. Readhead. Visualization diffusion tensor images of the mouse spinal cord. In *Proceedings of the 1998 IEEE Conference on Visualization*, pages 127–134, 1998. 29

[LC87] L Lorensen and H. Cline. Marching cubes : A high resolution 3d surface construction algorithm. In *Proceeding of ACM SIGGRAPH Conference*, pages 163–169, 1987. 27

[Lev88] Marc Levoy. Display of surfaces from surface data. *IEEE Computer Graphics and Applications*, 8(3) :29–37, 1988. 93

[Lev90] Marc Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3) :245–261, 1990. 93

[LHD+04] Robert S. Laramee, Helwig Hauser, Helmut Doleisch, Benjamin Vrolijk, Frits H. Post, and Daniel Weiskopf. The state of the art in flow visualisation : Dense and texture-based techniques. *Computer Graphics Forum (CGF)*, 23(2) :203–221, 2004. 27

[LHJ01] Eric Lamar, Bernd Hamann, and Kenneth I. Joy. A magnification lens for interactive volume visualization. In *In IEEE Pacific Conference on Computer Graphics and Applications*, pages 223–232, 2001. 67

[Lin00] Peter Lindstrom. Out-of-core simplification of large polygonal models. In *Computer Graphics Proceedings, ACM SIGGRAPH*, pages 259–262, 2000. 33, 82

[LLLV05] Marcos Lage, Thomas Lewiner, Hélio Lopes, and Luiz Velho. Chf : A scalable topological data structure for tetrahedral meshes. In *XVIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'05)*, pages 349–356, 2005. 24

[LM98] Scott Leutenegger and Kwan-Liu Ma. Fast retrieval of disk-resident unstructured volume data for visualization. In *Proceedings of the DIMACS Workshop on External Memory Algorithms and Visualization*, pages 279–290, 1998. 33

[Loh96] R. Lohner. Progress in grid generation via the advancing front technique. *Engineering with Computers*, 12 :186–201, 1996. 17

[LT97] H. Lopes and G. Tavares. Structural operators for modeling 3-manifolds. In *Solid Modeling and Applications*, pages 10–18, 1997. 24

[LTWH08] Ghu-Shi Li, Xavier Tricoche, Daniel Weiskopf, and Charles Hansen. Flow charts : Visualization of vector fields on arbitrary surfaces. In *IEEE transactions on visualization and computer graphics*, volume 14, pages 1067–1080, 2008. 29

[Luc92] Bruce Lucas. A scientific visualization renderer. In *VIS '92 : Proceedings of the 3rd conference on Visualization '92*, pages 227–234, 1992. 95

[Ma95] Kwan-Liu Ma. Parallel volume ray-casting for unstructured-grid data on distributed-memory architectures. In *PRS '95 : Proceedings of the IEEE symposium on Parallel rendering*, pages 23–30, 1995. 34

[Max95] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2) :99–108, 1995. 90, 92

[MC87] B. H. Mc Cormick. Visualization in scientific computing. In *Computer Graphics*, volume 21, November 1987. 16

[MC97] Kwan-Liu Ma and Thomas W. Crockett. A scalable parallel cell-projection volume rendering algorithm for three-dimensional unstructured data. In *PRS '97 : Proceedings of the IEEE symposium on Parallel rendering*, pages 95–ff., 1997. 34

[MCDF94] Steven Molnar, Michael Cox, Ellsworth David, and Henry Fuchs. A sorting classification of parallel rendering. *IEEE Computer Graphics and Applications*, 14(4) :23–32, 1994. 33

[Mic09a] Microsoft. *DirectX : SDK documentation*, 2009. http ://msdn.microsoft.com/. 31

[Mic09b] Microsoft. *Programming Guide for HLSL*, March 2009. http ://msdn.microsoft.com/. 32

[MJE91] R.S MacLeod, C.R Johnson, and P. R. Ershler. Construction of an inhomogenous model of the human torso to use in computational electrocardiography. In *IEEE Engineering in Medecine and Biology Society, 13th Annual International Conference*, pages 688–689, 1991. 35

[ML04] Ken Museth and Santiago Lombeyda. Tetsplat real-time rendering and volume clipping of large unstructured tetrahedral meshes. In *VIS '04 : Proceedings of the conference on Visualization '04*, pages 433–440, 2004. 96

[MMFE06] Ricardo Marroquim, Andre Maximo, Ricardo Farias, and Claudio Esperanca. Gpu-based cell projection for interactive volume rendering. *SIBGRAPI'06*, 0 :147–154, 2006. 96, 103, 104

[MP78] D. E. Muller and F. P. Preparata. Finding the intersection of two convex polyhedra. In *Theorical Computer Science*, volume 7, pages 217–236, 1978. 24

[MS06] Gerd Marmitt and Philipp Slusallek. Fast ray traversal of tetrahedral and hexahedral meshes for direct volume rendering. In *Eurographics/ IEEE-VGTC Symposium on Visualization*, pages 39–45, 2006. 94

[MTHG03] Oliver Mattausch, Thomas Theußl, Helwig Hauser, and Meister Eduard Gröller. Strategies for interactive exploration of 3d flow using evenly-spaced illuminated streamlines. Technical Report TR-186-2-03-04, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, April 2003. 68

[Möb27] August Ferdinand Möbius. Der barycentrische calcul. Technical report, Leipzig, 1827. 92

[NE04] Vijay Natarajan and Herbert Edelsbrunner. Simplification of three-dimensional density maps. In *IEEE Transactions on Visualization and Computer Graphics*, volume 10, pages 587–597, 2004. 34, 40, 41

[NNSM07] Michael B. Nielsen, Ola Nilsson, Andreas Söderström, and Ken Museth. Out-of-core and compressed level set methods. *ACM Trans. Graph.*, 26(4) :16, 2007. 82

[NVI07] NVIDIA. *NVIDIA CUDA : Programming Guide*, version 1.1 edition, 2007. 32

[Owe98] Steven J. Owen. A survey of unstructured mesh generation technology. In *7th International Meshing Roundtable*, pages 239–267, 1998. 17, 18

[Pas04] Valerio Pascucci. Isosurface computation made simple : Hardware acceleration, adaptive refinement and tetrahedral stripping. In *Proceedings of Eurographics, IEEE Symposium on Visualization*, pages 293–300, 2004. 27

[PF01] V. Pascucci and R. Frank. Global static indexing for real-time exploration of very large regular grids. In *Proceedings of SC 2001, High Performance Networking and Computing*, 2001. 33

[Pic91] André Pichot. *La Naissance de la Science. Tome 1 : Mésopotamie, Égypte.* Gallimard, 1991. 11

[PLB+01] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L.S. Avila, K.M. Raghu, R. Machiraju, and L. Jinho. The transfer function bake-off. *Computer Graphics and Application, IEEE*, 21(3) :16–22, 2001. 35

[PT94] Clifford A. Pickover and Stuart E. Tewksbury, editors. *Frontiers of Scientific Visualization*. John Wiley & Sons, Inc., 1994. 16

[Pup96] E. Puppo. Variable resolution terrain surfaces. In *Eight Canadian Conference on Computational Geometry*, pages 202–210, 1996. 45

[PVH+02] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and Doleisch H. Feature extraction and visualisation of flow fields. In *Eurographics 2002 State-of-the-Art Reports*, pages 69 –100, 2002. 27

[RB93] J Rossignac and P Borrel. Multi-resolution 3d approximations for rendering complex scenes. In B. Falcidieno and T. Kunii, editors, *Modeling in Computer Graphics : Methods and Applications*, pages 455–465, 1993. 39

[RS01] C. Rezk-Salama. *Volume Rendering Techniques for General Purpose Graphics Hardware*. PhD thesis, University of Erlangen-Nüremberg, 2001. 26

[RSHSG00] Christof Rezk-Salama, Peter Hastreiter, Jörg Scherer, and Günther Greiner. Automatic adjustment of transfer functions for 3d volume visualization. In *Proceedings Worshop Vision, Modeling and Visualization (VMV)*, pages 357–364, 2000. 35

[RYL+96] David M. Reed, Roni Yagel, Asish Law, Po-Wen Shin, and Naeem Shareef. Hardware assisted volume rendering of unstructured grids by incremental slicing. In *VVS '96 : Proceedings of the 1996 symposium on Volume visualization*, pages 55–64, 1996. 97

[Sai94] Takafumi Saito. Real-time previewing for volume visualization. In *VVS '94 : Proceedings of the 1994 symposium on Volume visualization*, pages 99–106, New York, NY, USA, 1994. ACM. 25

[SCCB05] C.T. Silva, J.L.D. Comba, S.P. Callahan, and F.F. Bernardon. A survey of gpu-based volume rendering of unstructured grids. In *Brazilian Journal Theoric and Applied Computing (RITA), Vol 12, Num 2*, pages 9 – 29, 2005. 93

[SCESL02] C. Silva, Y. Chiang, J. El-Sana, and P. Lindstrom. Out-of-core algorithms for visualization and computer graphics. In *IEEE Visualization'02, Course Notes for Tutorial 4*, 2002. 33, 82

[SEK+07] C.J. Stimpson, C.D. Ernst, P. Knupp, P.P. Pébay, and D. Thompson. *The VERDICT Library Reference Manual*. vtk.org, 2007. 22

[SG05] H. Si and K. Gaertner. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *the 14th International Meshing Roundtable*, pages 147–163, 2005. 50, 59

[SH00] P.M. Sutton and C.D. Hansen. Accelerated isosurface extraction in time-varying fields. *IEEE Transactions on Visualization and Computer Graphics*, 6(2) :98–107, 2000. 33

[Sha08] Ariel Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6) :1539–1556, 2008. 53

[She98] Jonathan Richard Shewchuck. Tetrahedral mesh generation by delaunay refinement. In *SCG'98 : Proceedings of the 14th annual Symposium on Computational Geometry*, pages 86–95, 1998. 17, 21, 23, 59

[She02] J. R. Shewchuck. Constrained delaunay tetrahedralizations and provably good boundary recovery. In *Eleventh International Meshing Roundtable*, pages 193–204, 2002. 24

[Si06] Hang Si. *TetGen : A Quality Tetrahedral Mesh Generator and Three-Dimensional Delauny Triangulator (User Manual)*, 2006. http ://tetgen.berlios.de. 50

[SJ08] Jason F. Shepherd and Chris R. Johnson. Hexahedral mesh generation constraints. *Engineering with Computers*, 24 :195 – 213, 2008. 17, 18

[SM97] Claudio T. Silva and Joseph S. B. Mitchell. The lazy sweep ray casting algorithm for rendering irregular grids. *IEEE Transactions on Visualization and Computer Graphics*, 3(2) :142–157, 1997. 97

[SML06] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit*. Kitware Inc., 4th edition edition, December 2006. 28

[SMW98] Claudio T. Silva, Joseph S. B. Mitchell, and Peter L. Williams. An exact interactive time visibility ordering algorithm for polyhedral cell complexes. In *VVS '98 : Proceedings of the 1998 IEEE symposium on Volume visualization*, pages 87–94, 1998. 95

[SP93] R. L. Sollenberg and Milgram P. Effects of stereoscopic and rotational displays in a 3d path tracing task. In *Human Factors*, pages 483–499, 1993. 87

[SR99] Andrzej Szymczak and Jarek Rossignac. Grow & fold : compression of tetrahedral meshes. In *SMA '99 : Proceedings of the fifth ACM symposium on Solid modeling and applications*, pages 54–64, 1999. 34

[SS05] Ralf Sondershaus and Wolfgang Strasser. View-dependent tetrahedral meshing and rendering. In *Proceedings of GRAPHITE '05*, pages 23–30, 2005. 45, 46, 47, 77

[SS06] Ralf Sondershaus and Wolfgang Strasser. Segment-based tetrahedral meshing and rendering. In *Proceedings of GRAPHITE '06*, pages 469–477, 2006. 45, 46, 47, 53, 77, 85

[ST90] P. Shirley and A. A. Tuchman. Polygonal approximation to direct scalar volume rendering. In *Proceedings San Diego Workshop on Volume Visualization, Computer Graphics*, volume 24, pages 63–70, 1990. 96, 97, 98, 99

[SWND07] David Shreiner, Mason Woo, Jackie Neider, and Tom Davis. *OpenGL Programming Guide : The Official Guide to Learning OpenGL, version 2.1*, 6th edition edition, 2007. 30

[SZ93] D. Silver and N. J. Zabusky. Quantifying visualizations for reduced modeling in nonlinear science : Extracting structures from data sets. *Journal of Visual Communication and Image Representation*, 4(1) :46–61, March 1993. 36

[SZL92] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26 :65–70, 1992. 49

[THJ99] I. J. Trotts, B. Hamann, and K. I. Joy. Simplification of tetrahedral meshes with error bounds. *IEEE Transactions on Visualization and Computer Graphics*, 5(3) :224–237, 1999. 34, 41

[THJW98] I. J. Trotts, B. Hamann, K. I. Joy, and D. F. Wiley. Simplification of tetrahedral meshes. In *Proceedings of IEEE VIS '98*, pages 287–295, 1998. 40

[TMFR05] J. Teran, Neil Molino, R. Fedkiw, and Bridson R. Adaptive physics based tetrahedral mesh generation using level sets. *Engineering with Computers*, 21(1) :2–18, 2005. 21

[TSC07] Natalya Tatarchuk, Jeremy Shopf, and DeCoro Christopher. Real-time isosurface extraction using the gpu programmable geometry pipeline. In *SIGGRAPH'07 : ACM SIGGRAPH Courses 2007*, pages 122–137, 2007. 27

[UBF+05] D. Uesu, L. Bavoil, S. Fleishman, J. Shepherd, and C.T. Silva. Simplification of unstructured tetrahedral meshes by point sampling. *4th International Workshop on Volume Graphics*, 0 :157–238, 2005. 39, 42, 48, 49, 50, 51, 59

[USM97] S. K. Ueng, C. Sikorski, and K.-L. Ma. Out-of-core streamline visualization on large unstructured meshes. *IEEE Transactions on Visualization and Computer Graphics*, 3(4) :370–380, 1997. 33

[VBHHar] Fabien Vivodtzev, Georges-Pierre Bonneau, Stefanie Hahmann, and Hans Hagen. Substructure topology preserving simplication of tetrahedral meshes. In *TopoInVis09*, To appear. 40

[VCL+07] Huy T. Vo, Steven P. Callahan, Peter Lindstrom, Valerio Pascucci, and Cláudio T. Silva. Streaming simplification of tetrahedral meshes. *Proceedings of IEEE Vis '07*, 13 :145–155, 2007. 42, 50, 83

[VCS+07] H.T. Vo, S.P. Callahan, N. Smith, C.T. Silva, W. Martin, D. Owen, and D. Weinstein. irun : interactive rendering of large unstructured grids. In *Eurographics Symposium on Parallel Graphics and Visualization*, pages 93–100, 2007. 33, 37

[VFSG06] Ivan Viola, Miquel Feixas, Mateu Sbert, and Eduard Gröller. Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics*, 12(5) :933–940, oct 2006. 36, 68

[VGVW99] A. Van Gelder, V. Verma, and J. Wilhelms. Volume decimation of irregular tetrahedral grids. *Computer Graphics International*, 0 :222, 1999. 41

[Viv05] Fabien Vivodtzev. *Hiérarchisation et Visualisation MultiRésolution de Résultats issus de Codes de Simulation*. PhD thesis, Université Grenoble I - Joseph Fourier, 2005. 40, 55, 56

[Wes90] Lee Westover. Footprint evaluation for volume rendering. In *SIGGRAPH '90 : Proceedings of the 17th annual conference on Computer Graphics and Interactive Techniques*, pages 367–376, 1990. 96

[Wil92] Peter L. Williams. Visibility-ordering meshed polyhedra. *ACM Trans. Graph.*, 11(2) :103–126, 1992. 94

[Wit01] Craig M. Wittenbrink. R-buffer : a pointerless a-buffer hardware architecture. In *HWWS '01 : Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 73–80, 2001. 95

[WKL99] David Weinstein, Gordon Kindlmann, and Eric Lundberg. Tensorlines : advection-diffusion based propagation through diffusion tensor fields. In *VIS'99 : Proceedings of the conference on Visualization'99*, pages 249–253, 1999. 29

[WKME03] M. Weiler, M. Kraus, M. Merz, and T. Ertl. Hardware-based ray casting for tetrahedral meshes. In *In Proc. Visualization '03*, pages 333–340, 2003. 94

[WMFC02] Brian Wylie, Kenneth Moreland, Lee Ann Fisk, and Patricia Crossno. Tetrahedral projection using vertex shaders. In *Proceedings of VVS '02*, pages 7–12, 2002. 96, 102, 104

[WMKE04] Manfred Weiler, Paula N. Mallon, Martin Kraus, and Thomas Ertl. Texture-encoded tetrahedral strips. In *VV '04 : Proceedings of the 2004 IEEE Symposium on Volume Visualization and Graphics (VV'04)*, pages 71–78, 2004. 94, 105

[WVGTG96] Jane Wilhelms, Allen Van Gelder, Paul Tarantino, and Jonathan Gibbs. Hierarchical and parallelizable direct volume rendering for irregular and multiple grids. In *VIS '96 : Proceedings of the 7th conference on Visualization '96*, pages 57–66, 1996. 97

[WZMK05] Lujin Wang, Ye Zhao, Klaus Mueller, and Arie Kaufman. The magic volume lens : An interactive focus+context technique for volume rendering. *Proceedings of IEEE VIS '05*, 0 :47, 2005. 68

[YS00] S. Yamakawa and K. Shimada. High quality anisotropic tetrahedral mesh generation via packing ellipsoidal bubbles. In *9th International Meshing RoundTable*, pages 263–273, 2000. 21

[YSGM04] Sung-Eui Yoon, Brian Salomon, Russell Gayle, and Dinesh Manocha. Quick-vdr : Interactive view-dependent rendering of massive models. In *VIS '04 : Proceedings of the conference on Visualization '04*, pages 131–138, 2004. 33

La simulation numérique génère des maillages de plus en plus gros pouvant atteindre plusieurs dizaines de millions de tétraèdres. Ces ensembles doivent être visuellement analysés afin d'acquérir des connaissances relatives aux données physiques simulées pour l'élaboration de conclusions. Les capacités de calcul utilisées pour la visualisation scientifique de telles données sont souvent inférieures à celles mises en œuvre pour les simulations numériques. L'exploration visuelle de ces ensembles massifs est ainsi difficilement interactive sur les stations de travail usuelles. Au sein de ce mémoire, nous proposons une nouvelle approche interactive pour l'exploration visuelle de maillages tétraédriques massifs pouvant atteindre plus de quarante millions de cellules. Elle s'inscrit pleinement dans le procédé de génération des simulations numériques, reposant sur deux maillages à résolution différente – un fin et un grossier – d'une même simulation. Une partition des sommets fins est extraite guidée par le maillage grossier permettant la reconstruction à la volée d'un maillage dit *birésolution*, mélange des deux résolutions initiales, à l'instar des méthodes multirésolution usuelles. L'implantation de cette extraction est détaillée au sein d'un processeur central, des nouvelles générations de cartes graphiques et en mémoire externe. Elles permettent d'obtenir des taux d'extraction inégalés par les précédentes approches. Afin de visualiser ce maillage, un nouvel algorithme de rendu volumique direct implanté entièrement sur carte graphique est proposé. Un certain nombre d'approximations sont réalisées et évaluées afin de garantir un affichage interactif des maillages birésolution.

Mots-clefs : Visualisation Scientifique, Maillages Tétraédriques, Interactivité, Ensembles de Données Massifs, Birésolution, Carte Graphique, Mémoire Externe, Rendu Volumique Direct.

Numerical simulations produce huger and huger meshes that can reach dozens of million tetrahedra. These datasets must be visually analyzed to understand the physical simulated phenomenon and draw conclusions. The computational power for scientific visualization of such datasets is often smaller than for numerical simulation. As a consequence, interactive exploration of massive meshes is barely achieved. In this document, we propose a new interactive method to interactively explore massive tetrahedral meshes with over forty million tetrahedra. This method is fully integrated into the simulation process, based on two meshes at different resolutions – one fine mesh and one coarse mesh – of the same simulation. A partition of the fine vertices is computed guided by the coarse mesh. It allows the on-the-fly extraction of a mesh, called *biresolution*, mixed of the two initial resolutions as in usual multiresolution approaches. The extraction of such meshes is carried out into the main memory (CPU), the last generation of graphics cards (GPU) and with an out-of-core algorithm. They guarantee extraction rates never reached in previous work. To visualize the biresolution meshes, a new direct volume rendering (DVR) algorithm is fully implemented into graphics cards. Approximations can be performed and are evaluated in order to guarantee an interactive rendering of any biresolution meshes.

Keywords : Scientific Visualization, Tetrahedral Meshes, Interactivity, Massive Datasets, Biresolution, Graphics Card, Out-of-core Implementation, Direct Volume Rendering.