Visualisation de données volumiques massives : application aux données sismiques

Laurent Castanie

Institut National Polytechnique de Lorraine

École Nationale Supérieure de Géologie **École doctorale RP2E**

Visualisation de Données Volumiques Massives

Applications aux Données Sismiques

THÈSE

présentée et soutenue publiquement le 24 Novembre 2006

pour l'obtention du

Doctorat de l'Institut National Polytechnique de Lorraine

Spécialité Géosciences

par

Laurent Castanié

Composition du jury

Rapporteurs :	Charles D. Hansen Helmut Schaeben
Examinateur :	Georges-Pierre Bonneau
Invités :	Fabien Bosquet Xavier Cavin
Directeur :	Jean-Laurent Mallet
Co-Directeur :	Bruno Lévy

Centre de Recherches Pétrographiques et Géochimiques
Laboratoire d'Infographie et d'Analyse de Données
Rue du Doyen Marcel Roubault - 54500 Vandœuvre

Mis en page avec la classe thloria.

Remerciements

Je tiens en premier lieu à remercier Jean-Laurent Mallet, mon directeur de thèse, qui en infatigable initiateur et animateur du projet Gocad nous permet aujourd'hui d'étudier et de travailler dans cet environnement stimulant. Merci également à Bruno Lévy qui m'a accordé sa confiance il y a trois ans en acceptant de co-diriger ma thèse malgré la charge de travail déjà conséquente que représentait la création du projet ALICE à l'INRIA Lorraine. Il a été tout au long de ces trois années une source intarissable de conseils, notamment pour la rédaction des articles et du mémoire, et a su chasser mes moments de doute. Cette thèse n'aurait pas vu le jour et ne serait pas ce qu'elle est sans le soutien continu de Fabien Bosquet de la société Earth Decision (aujourd'hui entité de la société Paradigm) et son travail remarquable dans l'architecture de VolumeExplorer. Je lui suis extrêmement reconnaissant de la touche personnelle qu'il a su apporter pour valoriser chacune des réalisations de mes travaux. Ses multiples conseils ont également guidé mes pas dans la société, sachant doser les impératifs industriels pour m'impliquer sans jamais entraver l'avancement de la thèse. Je remercie aussi à ce sujet Jean-Claude Dulac, président de Earth Decision, qui a d'autre part accepté de financer ces travaux. Sur l'encadrement de cette thèse, je tiens également à remercier Xavier Cavin qui a accueilli un "géologue" dans son équipe et sans qui cette thèse n'aurait pas de chapitre 6. J'en profite pour souhaiter une longue vie à DViz et à sa collaboration avec Paradigm.

Je suis profondément reconnaissant envers Charles D. Hansen, professeur à l'Université d'Utah, et Helmut Schaeben, professeur à l'Université de Freiberg, qui malgré leurs nombreuses obligations ont accepté de rédiger un rapport sur cette thèse et m'ont fait l'honneur de leur présence. Merci également à Georges-Pierre Bonneau, professeur à l'Université Joseph Fourier de Grenoble, pour avoir accepté de faire partie de mon jury.

Au cours de ces trois années, j'ai eu simultanément trois bureaux : l'un chez Earth Decision, l'autre au LIAD et enfin un dernier à l'INRIA. Je tiens donc à remercier ceux qui ont su sortir leurs griffes pour s'assurer que je dispose d'une place quelque soit le lieu (ils se reconnaîtront).

Ceci me conduit tout naturellement à m'excuser auprès des gens du LIAD qui, malgré mes absences prolongées, m'ont toujours accordé un accueil chaleureux. Merci donc aux thésards, présents et passés, que j'ai pu côtoyer : Sarah, Manu F., Laeti, Pierre M., Pierre K., Luc, Tobias, Marco, Anlor, Bruno, et les plus anciens François, David, Laurent L., Laurent S. et Rémi. Merci également à Guillaume qui, avec François, m'ont mis le pied à l'étrier il y a de cela quelques années à l'ENSG et m'ont communiqué l'envie de continuer sur cette voie, à Christian dont les conseils avisés n'ont d'égal que l'humour, à Sophie partie vers d'autres horizons, à Pierre J., Jean-Jacques et Mme Cugurno pour son soutien logistique.

Pour m'avoir accueilli dans leur bonne humeur durant ma dernière année de thèse, je remercie les membres de l'équipe ALICE de l'INRIA Lorraine : Bruno, Blix, Bruninho, Greuh, Nico, Laurent, Isa avec une pensée particulière pour les "DViz guys", à savoir Xa, Butch qui a réalisé un véritable travail d'orfèvre sur DViz (y'a pas d'magie !) et pour son aide sur le DHCS, Alm, Olive et Tibur. Pardonnez encore l'inculture d'un géologue et merci d'avoir comblé nombreuses de mes lacunes par vos enseignements avisés sur le matériel.

Je réserve un grand merci pour les employés de Earth Decision Nancy où j'ai passé la plus grande partie de mon temps durant les trois dernières années. Cela aura été un grand plaisir d'aller travailler tous les jours dans la convivialité. En essayant de n'oublier personne, je remercie Fabien, Manu L. pour ses conseils et certaines figures du mémoire, Joël qui m'a fait vivre ses courses par procuration durant les longs mois de rédaction, Louis toujours disponible pour

débloquer une situation, Thierry, Didier, Stéphane, Olivier M., Olivier G., Matthieu, Etienne (une page de remerciements, c'est déjà beaucoup trop de papier !), Elsa, Magali, Isabelle, Richard qui m'a été d'un grand secours sous Linux, Tof, Sébastien, Thibault, Rémy, Christophe P., Delphine, Arnaud, Sabrina, Bernadette, Elahe, Cédric, Damien et Anne-Sophie. Je remercie également Isa et Alice de Paris pour leur sympathie, ainsi que Stan de Houston pour m'avoir guidé dans mes choix et Susan sans qui l'article de First Break n'aurait pas été soumis.

Je finirai cette longue liste de remerciements par ceux sans qui je n'aurais jamais accompli ce travail : mes parents qui, depuis le début, m'ont toujours soutenu dans mes choix et ont su constamment trouver les mots qu'il fallait pour ôter mes doutes, mon frère dont l'attention et le soutien dépassent les mots, et enfin Christelle qui a su supporter mes humeurs, la distance et les longs mois de rédaction. Elle m'a soutenu tout au long de ces trois années jusqu'à affronter la relecture complète du manuscrit. Pour tout cela, je lui dis un grand merci.

À Christelle,
à mes parents,
à mon grand-père.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Le travail présenté dans ce mémoire s'insère dans le domaine de la visualisation scientifique, formalisé en 1987 aux États-Unis dans un rapport de la National Science Foundation (N.S.F.) [McCormick B.H. *et al.*, 1987]. Cette discipline s'attache à mettre en évidence des informations dans un volume conséquent de données issues soit de l'acquisition, soit de simulations. Les problèmes traités relèvent donc à la fois du domaine de la visualisation informatique et de la discipline scientifique concernée. En effet, outre la nécessité de développer des techniques informatiques génériques pour la visualisation de données, un élément essentiel dans toute problématique de visualisation scientifique est la nécessité de prendre en compte la spécificité de la discipline scientifique concernée. Dans le cadre de ce mémoire, nous nous intéressons à la visualisation de données issues de la géophysique. Cette étude a été financée par la société Earth Decision[1] et s'inscrit dans le cadre du projet Gocad[2] de l'École Nationale Supérieure de Géologie de Nancy, en coopération avec le projet ALICE[3] de l'INRIA Lorraine. Le projet Gocad, initié à la fin des années 1980, vise à définir un cadre mathématique et informatique pour le traitement de l'ensemble des problèmes rencontrés dans la modélisation des phénomènes en géosciences, également appelée *géomodélisation*. Le résultat de cette recherche est développé et commercialisé par la société Earth Decision essentiellement pour l'industrie du pétrole et du gaz naturel, fournissant ainsi une suite logicielle qui couvre l'ensemble des métiers de la chaîne de traitement en exploration-production. Cette chaîne de traitement a pour finalité d'optimiser l'extraction des hydrocarbures par la prédiction des volumes de production à partir d'un modèle numérique tridimensionnel du sous-sol. Le projet INRIA ALICE traite de problèmes fondamentaux de l'informatique graphique, dont l'algorithmique géométrique et la simulation des interactions lumineuses. Pour étendre le résultat de ces recherches fondamentales à des jeux de données à plus grande échelle, intéressants d'un point de vue industriel, un axe de recherche connexe dans le projet concerne, en relation avec le CRVHP[4], la visualisation haute-performance, impliquant notamment l'utilisation de clusters de PCs pour afficher des modèles massifs. Cette compétence s'est avérée essentielle dans le cadre de nos travaux de visualisation scientifique.

Problématique

Les données géophysiques se situent à l'amorce de la chaîne d'exploration-production avec l'acquisition des données sismiques. En enregistrant le temps de parcours dans le sol d'ondes acoustiques générées artificiellement en surface et réfléchies par les interfaces géologiques, il est possible de reconstruire après traitement une image tridimensionnelle du sous-sol : l'*image sis-*

[1] http ://www.earthdecision.com (entité de Paradigm depuis Août 2006, http ://www.paradigmgeo.com)
[2] http ://www.gocad.org
[3] http ://alice.loria.fr
[4] Calcul Réseau Visualisation Haute Performance, http ://crvhp.loria.fr

mique. L'*interprétation sismique*, fondée sur ces données, fournit une information capitale sur la géométrie des interfaces géologiques, ainsi que sur les propriétés pétrophysiques des roches en présence. Outre les algorithmes métiers permettant de construire un modèle géologique tridimensionnel à partir de cette image, il est nécessaire de disposer d'outils de visualisation adaptés au problème traité. Tandis que le développement d'algorithmes métiers a récemment été exploré dans le cadre du projet Gocad [Labrunye E., 2004], l'aspect visualisation a été abordé d'un point de vue purement industriel à la fin des années 1990 avec VolumeExplorer [Bosquet F. et Dulac J.C., 2000], le module de visualisation de volumes sismiques du logiciel Gocad. Les grandes avancées technologiques dont a bénéficié le matériel graphique au début des années 2000, notamment sous l'impulsion de l'industrie du jeu vidéo, n'ont pas été exploitées dans le cadre du projet Gocad, et de manière marginale dans les géosciences en général. L'objectif de notre travail est de tirer profit des nouvelles fonctionnalités offertes par le matériel graphique afin de fournir des outils de visualisation des données sismiques plus performants que les outils classiques [Castanié L. *et al.*, 2005a, Castanié L. *et al.*, 2005b, Castanié L. *et al.*, 2005c]. Cet objectif est double, visant d'une part à améliorer la qualité de l'image afin de la rendre "plus interprétable" et d'autre part à fournir de nouvelles fonctionnalités inaccessibles avec les outils classiques.

De plus, les avancées technologiques associées aux capteurs géophysiques utilisés en acquisition ainsi qu'aux serveurs de traitement ont respectivement permis au fil des années d'acquérir des données de plus haute résolution sur de plus grandes zones et de traiter des volumes de plus en plus grands. Par voie de conséquence, les volumes à interpréter et donc à visualiser ont subi la même évolution. L'imagerie sismique en exploration pétrolière atteint aujourd'hui une résolution de l'ordre de la dizaine de mètres sur plusieurs kilomètres de profondeur pour un bassin couvrant plusieurs dizaines de milliers de kilomètres carrés en surface [Brown A.R., 2004, Dopkin D. et James H., 2006]. Il est ainsi relativement courant de devoir visualiser des volumes de plusieurs dizaines de gigaoctets de données sismiques, voire dans certains cas de plusieurs centaines. Du point de vue des outils de visualisation, les difficultés qui en découlent sont évidentes. Le matériel standard mis à disposition du géophysicien et/ou du géologue chargé de l'interprétation est limité et rapidement dépassé par une telle évolution. Ainsi, le deuxième aspect de notre travail a consisté à développer des algorithmes permettant de prendre en compte les limitations associées au matériel pour améliorer l'interactivité lors de l'interprétation de volumes sismiques de grande taille [Castanié L. *et al.*, 2005c, Castanié L. *et al.*, 2006].

Contributions

Tout au long de ce travail, nous nous sommes efforcés de garder à l'esprit le texte référence de Chris Johnson sur les problèmes les plus importants en visualisation scientifique [Johnson C., 2004]. Cet article non seulement dresse une liste de problèmes ouverts, mais se veut également le porte-parole d'une démarche concertée servant à consolider les bases d'une discipline en pleine maturation. Parmi les quinze points qu'il énonce, sept d'entre eux illustrent particulièrement bien la démarche que nous avons appliquée dans notre travail ainsi que nos contributions :

Penser en termes d'application ("Think about the science") : Il s'agit ici de mettre l'accent sur la nécessaire synergie entre spécialistes du domaine d'application concerné et spécialistes de la visualisation de sorte que les solutions de visualisation proposées prennent correctement en compte les contraintes propres au domaine scientifique d'application. C'est en effet en nous ef-

forçant de garder une position neutre à la frontière entre le domaine de l'interprétation sismique et celui de la visualisation que nous avons mené nos travaux, lesquels ont ainsi pu être validés dans les deux communautés scientifiques, respectivement dans [Castanié L. *et al.*, 2005a, Castanié L. *et al.*, 2005b] et dans [Castanié L. *et al.*, 2005c, Castanié L. *et al.*, 2006].

Quantifier la pertinence ("Quantify effectiveness[5]") : L'objectif est d'appliquer la méthode scientifique dans la mise au point de techniques de visualisation pour une application précise. En résumé, celle-ci consiste à observer et décrire dans un premier temps afin de proposer une hypothèse explicative, laquelle sert ensuite à prédire de nouvelles observations afin de valider la pertinence de la technique employée. C'est précisément la démarche que nous avons suivie, illustrée dans le Chapitre 3. Nous y analysons en effet la pertinence des techniques de visualisation volumique classiques dans le contexte particulier des données sismiques, ce qui nous permet d'identifier leurs limitations et d'appliquer avec succès des techniques issues d'autres domaines d'applications tels que l'imagerie médicale.

Visualisation multimodale ("Multifield visualization") : De manière stricte, la visualisation multimodale concerne la combinaison de plusieurs volumes d'information dans la démarche d'interprétation. Nous abordons le problème au Chapitre 4 en considérant plus largement toutes les possibilités de combinaison de l'information, qu'elle soit d'origine diverse (sismique ou structurale) ou même de nature diverse (volumique ou surfacique). Ceci nous permet entre autres d'étendre la notion de multimodalité en ré-injectant des données structurales issues des premières étapes de l'interprétation dont la combinaison avec l'information sismique permet de converger plus rapidement vers un résultat.

Interaction homme-machine ("Human-computer interaction") : Ce point concerne l'amélioration des processus d'interaction entre l'homme et l'ordinateur, notamment par la mise au point d'interfaces utilisateur adéquates. Il s'agit là d'une discipline à part entière à laquelle notre travail ne se rapporte pas directement. Cependant, le Chapitre 4 sur la visualisation sismique multimodale détaille les étapes de construction d'une interface multimodale générique extensible permettant de combiner plusieurs volumes d'information de manière extrêmement flexible.

Environnements intégrés de travail ("Integrated problem-solving environments (PSEs)") : La notion d'interaction homme-machine est ici renforcée dans le but de fournir à l'utilisateur un environnement de travail unique pour l'ensemble des étapes de l'interprétation. L'interprétation des données sismiques s'insère à plus grande échelle dans le processus de modélisation du sous-sol en exploration-production pétrolière. Dans ce contexte, la démarche d'intégration a prévalu dès les premiers développements du projet Gocad avec l'implantation d'une plate-forme fédératrice autour de laquelle viennent se greffer des modules spécialisés. Du point de vue des réalisations logicielles, notre travail a été implanté sous la forme de classes C++ dans le module VolumeExplorer destiné à l'interprétation des données sismiques en général, et notamment à la visualisation des données volumiques. C'est précisément l'existence de cet environnement intégré qui nous permet, au Chapitre 5, de valoriser un système centralisé de gestion de la mémoire pour le couplage de la visualisation et des calculs.

Visualisation globale/locale (les détails dans leur contexte) ("Global/local visualization (details within context)") : L'interprétation de données implique une démarche de détail dans un contexte plus large, et ce quel que soit le domaine d'application. Ceci est particulièrement vrai pour l'interprétation des données sismiques qui implique différentes échelles de raisonnement depuis le bassin sédimentaire dans son ensemble (plusieurs dizaines de kilomètres)

[5]Le terme "effectiveness" fait référence ici à l'efficacité globale par rapport au domaine d'application, et non pas au temps d'exécution.

jusqu'au réservoir localisé (plusieurs centaines de mètres). Il est donc important, tout en identifiant des événements locaux, de ne pas perdre de vue le contexte global de l'interprétation. L'environnement intégré que fournit le logiciel Gocad participe à cet effort de cohérence, les données volumiques coexistant avec d'autres types de données telles que des données de forage, des surfaces géologiques extraites du cube sismique ou encore des tracés de puits de production, le tout pouvant être visualisé de manière simultanée. C'est donc dans ce contexte que nous avons implanté nos algorithmes. D'autre part, nous insistons au Chapitre 6 sur l'importance de pouvoir naviguer entre les différentes échelles d'interprétation avec un système capable de visualiser et de déplacer en temps réel une sonde de visualisation volumique de l'ordre du gigaoctet de données dans un volume à l'échelle du bassin sédimentaire de l'ordre de la centaine de gigaoctets.

Exploiter au mieux les architectures matérielles ("Efficiently utilizing novel hardware architectures") : Finalement, ce dernier point concerne la mise au point d'algorithmes et de systèmes capables de tirer le meilleur parti des ressources mises à disposition, notamment du matériel graphique. Ce dernier expose régulièrement de nouvelles fonctionnalités qu'il convient d'exploiter de manière appropriée. Les systèmes de visualisation présentés dans les Chapitres 3 et 4, respectivement sur la visualisation volumique du signal sismique et sur la visualisation multimodale en sismique, sont des exemples de solutions apportées à des problèmes concrets par l'utilisation adéquate des fonctionnalités exposées par le matériel graphique. Johnson aborde également l'utilisation de clusters graphiques qui permettent d'augmenter significativement la puissance de traitement disponible dans le cas de grandes quantités de données. C'est le sujet du Chapitre 6 sur le passage d'une échelle d'interprétation locale, légèrement supérieure au réservoir, à une échelle régionale, englobant le bassin sédimentaire dans son ensemble, qui implique des volumes de données de l'ordre de la centaine de gigaoctets et nécessite une puissance de calcul qui dépasse largement celle d'une seule station de travail.

Organisation du mémoire

Le mémoire est organisé en trois parties. La Partie I présente le contexte de notre travail, tant du point de vue de l'application, l'interprétation sismique, que du point de vue scientifique, la visualisation volumique. Le Chapitre 1 permet de situer l'interprétation sismique dans la chaîne d'exploration-production et d'en comprendre ainsi les principaux enjeux. Le Chapitre 2 réalise un état de l'art des principales techniques de visualisation de données volumiques.

Après ces chapitres introductifs, la Partie II se place dans le contexte de la station d'interprétation pour aborder les différents aspects de notre travail. Le premier, détaillé au Chapitre 3, concerne la mise au point et l'évaluation d'une technique de visualisation volumique adaptée aux particularités des données sismiques. Le Chapitre 4 sur la visualisation sismique multimodale aborde un problème plus général de l'interprétation sismique qui concerne la mise en relation de plusieurs sources d'information au cours du processus d'interprétation. Pour terminer sur cette partie, le Chapitre 5 s'intéresse au problème de la taille des données et décrit un système de gestion de la mémoire destiné au couplage de la visualisation et des calculs.

Finalement, la Partie III permet de porter certains des concepts de visualisation implantés dans le cadre de la station de travail vers les clusters graphiques qui exposent une puissance de traitement bien supérieure. Du point de vue de l'interprétation, le Chapitre 6 propose un système de visualisation qui permet de manipuler des volumes de données dépassant les capacités d'une seule station et donc d'appréhender un bassin sédimentaire dans son ensemble avec une réelle démarche d'interprétation à échelle régionale.

Première partie

Contexte de l'étude

Chapitre 1

Contexte applicatif : les données sismiques et leur interprétation

Ce chapitre a pour but de replacer notre travail dans son contexte applicatif, à savoir l'interprétation des données sismiques. Nous allons tout d'abord nous intéresser à l'origine des données : leur mode d'acquisition et de traitement pour aboutir à l'image sismique, objet de l'interprétation. Nous replacerons ensuite l'interprétation sismique dans le contexte plus général de l'exploration-production et de l'extraction des hydrocarbures. Finalement, nous aborderons les points essentiels de la visualisation des données sismiques et de leurs spécificités (forte fréquence spatiale le long de la dimension verticale, bruit, ...), lesquelles orienteront la suite du travail.

1.1 De la propagation des ondes acoustiques à l'image sismique

La construction de l'image sismique tridimensionnelle est un processus complexe, analogue à une échographie du sous-sol (Figure 1.3), qui fait intervenir des notions physiques et mathématiques dépassant le cadre de notre exposé. Nous allons cependant en décrire les principes de base afin de mettre en relief quelques points essentiels concernant la nature et le format des données que nous serons amenés à manipuler. Pour plus de détails concernant les fondements théoriques de la propagation des ondes acoustiques et de l'utilisation qui en est faite en exploration-production pour la construction de l'image sismique tridimensionnelle, le lecteur pourra se référer à des ouvrages plus complets tels que [Yilmaz O., 1987] ou [Robein E., 1999].

1.1.1 Théorie de la propagation des ondes acoustiques en milieu élastique

Sous l'effet de contraintes extérieures faibles, le sous-sol subit des déformations linéaires et réversibles dites *élastiques*. Dans un modèle de déformation élastique, les particules d'un milieu soumis à une contrainte extérieure faible entrent en mouvement vibratoire autour de leur position initiale. Du fait des contraintes liées au milieu environnant, les particules en mouvement se compriment modifiant ainsi localement l'état de pression du milieu. Ce mouvement de particules et la variation locale de l'état de pression qui en découle se propagent aux particules voisines. L'état vibratoire en fonction du temps et de l'espace, qui se définit indifféremment par le déplacement d'une particule par rapport à sa position initiale ou par la variation locale de l'état de

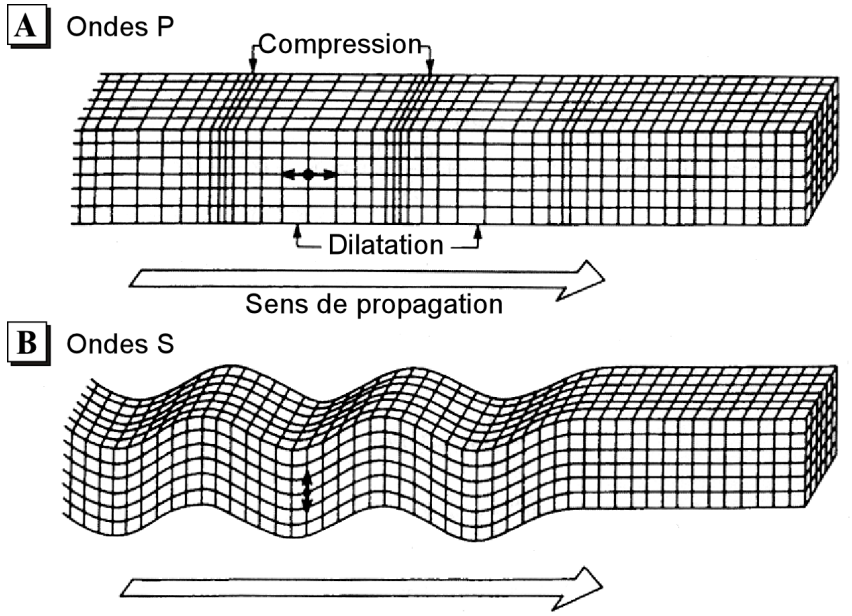

FIG. 1.1 – Modes vibratoires des particules dans la propagation des ondes acoustiques. (A) Ondes de compression transmettant un mouvement radial dans le sens du front d'onde ; (B) Ondes de cisaillement transmettant un mouvement tangentiel au front d'onde (d'après [BOLT B., 1982]).

pression, est appelé *champ d'onde*. D'autre part, la limite à un instant t entre les particules au repos et les particules atteintes par l'onde est appelée *front d'onde*.

De cette manière et sous l'effet d'une source d'énergie acoustique positionnée en un point de la surface terrestre, se propagent dans le sous-sol les fronts d'ondes acoustiques. Il existe deux types d'ondes selon le mode de transmission de l'énergie aux particules voisines. Celui-ci est directement lié au mode vibratoire des particules (Figure 1.1) :

- les ondes P (de l'anglais "push"), dites de compression, transmettent un mouvement radial dans le sens du front d'onde et sont les plus rapides ;
- les ondes S (de l'anglais "shake"), dites de cisaillement, transmettent un mouvement tangentiel au front d'onde. En l'absence de rigidité du milieu (e.g. milieu liquide), l'énergie de ces dernières ne peut se propager.

La réponse vibratoire du milieu à la surpression induite par la source dépend de son *impédance acoustique* I. Cette dernière est définie comme suit :

$$I = \rho \cdot V \tag{1.1}$$

où ρ est la masse volumique du milieu et V la vitesse de propagation (ou célérité) des ondes acoustiques dans ce dernier. Plus l'impédance acoustique I d'un milieu est forte, plus faible est la vitesse de particule pour une surpression donnée.

À l'interface entre deux milieux d'impédance acoustique différente (Figure 1.2), une partie de l'énergie de l'onde incidente est transmise au milieu adjacent sous la forme d'une onde réfractée tandis que l'autre partie est réfléchie dans le milieu courant, générant ainsi un front d'onde réfléchi. La part d'énergie réfléchie (resp. réfractée) est déterminée par le ratio entre l'amplitude de l'onde réfléchie (resp. réfractée) et celle de l'onde incidente. Ce ratio, noté C_r (resp. C_t), est appelé *réflectivité/coefficient de réflexion* (resp. *transmissibilité/coefficient de transmission*) et

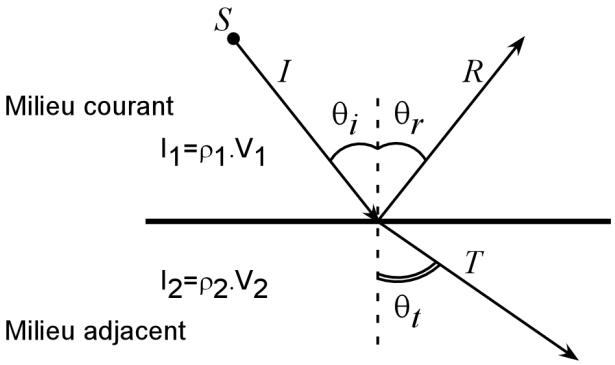

FIG. 1.2 – Géométrie des rayons incident I, réfléchi R et réfracté T à l'interface entre deux milieux adjacents d'impédances acoustiques respectivement I_1 et I_2 différentes. ρ_1 et ρ_2 sont respectivement les densités volumiques, et V_1 et V_2 les vitesses de propagation des ondes acoustiques dans les milieux 1 et 2 ($V_1 < V_2$). S est la source d'énergie acoustique.

dépend du contraste d'impédance acoustique entre les deux milieux :

$$\begin{aligned} C_r &= \frac{A_r}{A_i} = \frac{I_2 - I_1}{I_2 + I_1} \\ C_t &= \frac{A_t}{A_i} = \frac{2 \cdot I_1}{I_2 + I_1} \end{aligned} \tag{1.2}$$

où A_i, A_r et A_t sont respectivement les amplitudes des ondes incidente, réfléchie et réfractée, et I_1 et I_2 les impédances acoustiques des milieux courant et adjacent. Notons que, dans le cas de la sismique, la réflectivité est généralement faible, de l'ordre de 0,05-0,1 en valeur absolue, ce qui implique que la proportion relative de l'énergie réfléchie est faible. Pour des valeurs de réflectivité de +0,2 ou -0,2, l'énergie réfléchie sera très importante. C'est le cas par exemple de la frontière entre un sable poreux et un grès dense qui constitue une surface de réflexion proéminente. À l'inverse, l'interface entre deux formations argileuses d'impédance acoustique similaire réfléchit une faible proportion de l'énergie incidente.

Les lois de réflexion-réfraction qui gouvernent la transmission de l'énergie entre deux milieux adjacents de propriétés acoustiques différentes sont identiques à celles qui gouvernent la transmission de la lumière. Ainsi il est plus intuitif de considérer la transmission des ondes à travers une interface du point de vue de la géométrie de leurs rayons, c'est-à-dire des lignes perpendiculaires aux fronts d'ondes successifs. Considérant un rayon incident I issu d'une source S en un point d'une discontinuité acoustique (Figure 1.2), la géométrie des rayons réfléchi R et réfracté T est déterminée par la loi de Snell-Descartes :

$$\begin{aligned} \theta_i &= \theta_r \\ \frac{\sin\theta_i}{V_1} &= \frac{\sin\theta_t}{V_2} \end{aligned} \tag{1.3}$$

où θ_i, θ_r et θ_t sont respectivement les angles que forment localement les rayons incident, réfléchi et réfracté avec la normale à la surface de réflexion-réfraction, et V_1 et V_2 les vitesses de propagations des ondes acoustiques dans les milieux courant et adjacent.

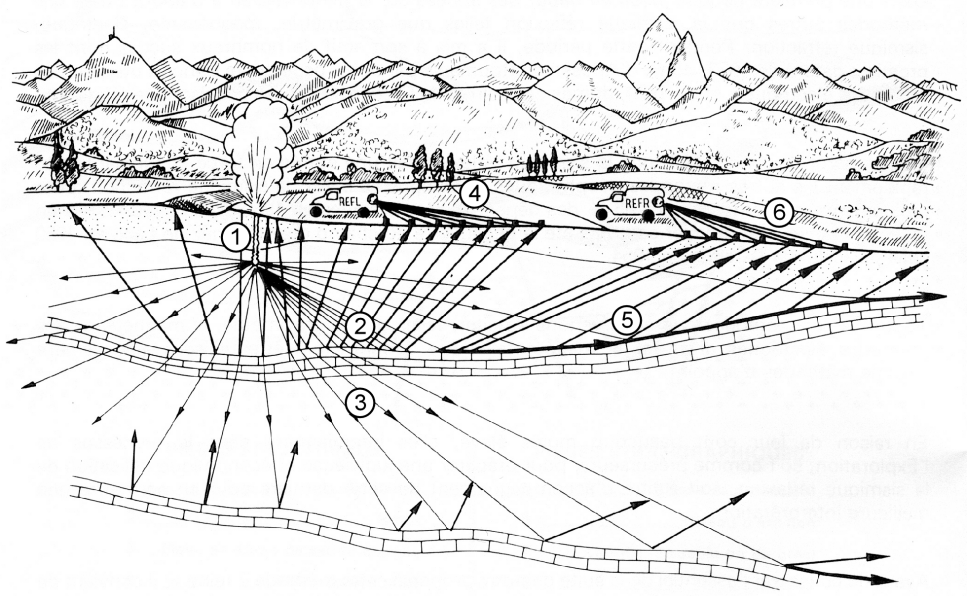

FIG. 1.3 – Principe de l'acquisition en sismique terrestre : cas de la *sismique réflexion* et de la *sismique réfraction*. (1) Source d'énergie acoustique par explosion en surface ; (2) Ondes réfléchies ; (3) Ondes réfractées non coniques ; (4) Géophones pour l'enregistrement des ondes réfléchies en sismique réflexion ; (5) Ondes réfractées coniques ; (6) Géophones pour l'enregistrement des ondes réfractées coniques en sismique réfraction (Image Total, d'après [LABRUNYE E., 2004]).

1.1.2 Acquisition de l'amplitude sismique : la sismique réflexion

L'acquisition sismique débute par l'émission d'énergie acoustique en surface, soit sur le sol en sismique terrestre, soit dans l'eau en sismique marine. Les ondes acoustiques ainsi générées se propagent dans les différentes couches géologiques suivant les lois énoncées au paragraphe précédent : elles subissent des réflexions/réfractions à l'interface entre deux couches aux propriétés différentes, comme un sable poreux et un grès dense. L'enregistrement en surface des temps d'arrivée des échos permet de déterminer la profondeur des interfaces successivement rencontrées. La *sismique réflexion* qui enregistre les ondes réfléchies se distingue de la *sismique réfraction* qui enregistre un type particulier d'ondes réfractées se propageant le long des interfaces et appelées *ondes coniques* (Figure 1.3). Tandis que la sismique réflexion intervient en tant qu'outil d'imagerie structurale du sous-sol, la sismique réfraction fournit essentiellement des informations sur la vitesse de propagation des ondes acoustiques dans les différentes formations traversées. La méthode la plus communément utilisée en exploration-production est la sismique réflexion, tandis que la sismique réfraction reste d'utilisation marginale. Bien que générant à la fois des ondes P et S, l'acquisition sismique n'enregistre que les ondes P, dites *primaires* du fait de leur vitesse de propagation plus grande et donc de leur arrivée précoce par rapport aux ondes S.

Le matériel utilisé en acquisition sismique consiste en une ou plusieurs sources d'énergie acoustique, une série de capteurs et un laboratoire d'enregistrement. Selon l'environnement considéré, terrestre ou marin, les modes de mise en œuvre et le matériel utilisé diffèrent. Par exemple, la source en sismique terrestre est placée à même le sol qu'elle ébranle directement, un explosif ou un camion vibrateur faisant l'affaire. Dans le cas de la sismique marine, la source la plus couramment utilisée est un canon à air placé dans l'eau. Celui-ci libère une bulle d'air comprimé provocant ainsi une surpression qui se propage jusqu'au sol marin et pénètre dans le sous-sol.

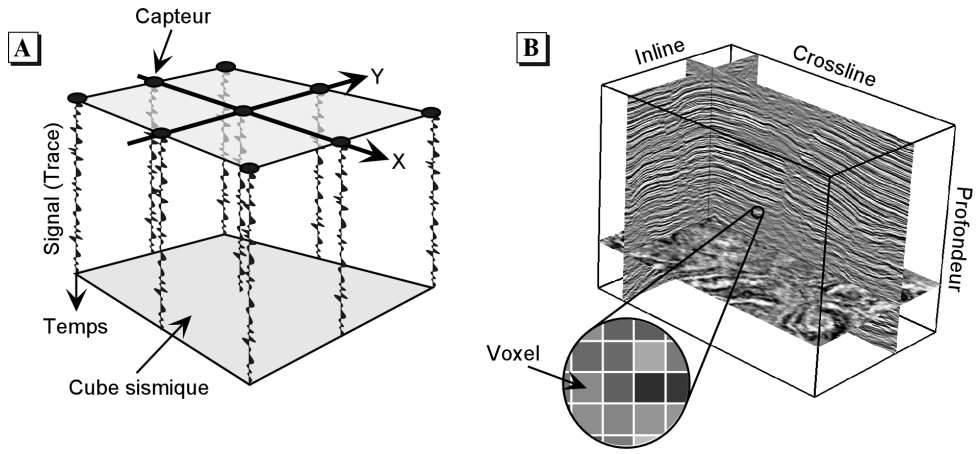

FIG. 1.4 – De l'enregistrement à intervalles réguliers de signaux sismiques dans l'espace et dans le temps (A) à l'image sismique tridimensionnelle (B) (d'après [LABRUNYE E., 2004]).

Du point de vue des capteurs, la différence est du même ordre. La sismique terrestre utilise des géophones couplés aux particules du sol. Ceux-ci émettent un signal proportionnel non pas à l'amplitude de leur déplacement mais à leur vitesse de déplacement, directement liée à celle des particules. En sismique marine, des hydrophones émettent un signal directement proportionnel aux écarts de pression mesurés dans l'eau. Le signal électrique émis par les capteurs, ou *signal sismique*, est enregistré sur bande dans le laboratoire et constitue l'*amplitude sismique*. Il s'agit donc d'une grandeur physique proportionnelle à la mesure de vitesse (sismique terrestre) ou de pression (sismique marine) au niveau des capteurs. Le coefficient de proportionnalité n'étant pas enregistré, l'amplitude sismique est une grandeur relative. Notons que l'enregistrement sur bande du signal sismique est effectué en continu. Cependant, son stockage sous forme numérique implique un échantillonnage dans le temps. Les valeurs d'amplitude sont donc stockées à pas de temps réguliers le plus généralement de 4 ms, ce qui permet d'échantillonner sans perte d'information la bande fréquentielle sismique traditionnelle, dite "haute fréquence" et qui se situe entre 8 et 80 Hz.

1.1.3 Traitement du cube sismique : construction de l'image sismique

L'enregistrement numérique sur bande échantillonne dans le temps le signal sismique émis au niveau de chaque capteur. Les capteurs étant disposés géographiquement à intervalles réguliers, il est possible de représenter les valeurs d'amplitude sismique dans une *grille régulière* (Section 2.1), le *cube sismique*, dont les axes horizontaux correspondent au quadrillage géographique des capteurs et l'axe vertical au temps (Figure 1.4-A). Ainsi, chaque colonne du cube sismique correspond à l'enregistrement dans le temps du signal sismique au niveau d'un capteur donné. L'ensemble des valeurs d'amplitude sismique ainsi échantillonnées verticalement constitue une *trace sismique*.

Le cube sismique ainsi obtenu ne constitue pas une image à proprement parler du sous-sol. En effet, il s'agit seulement d'un enregistrement dans le temps et l'espace du signal sismique. Une amplitude sismique observée au niveau d'un capteur donné en surface correspond rarement à un point situé à la verticale de ce capteur. Le *traitement sismique* fait référence à l'ensemble des méthodes qui permettent de transformer le signal sismique brut en une image tridimensionnelle du sous-sol communément appelée l'image sismique. Les techniques d'imagerie en sismique réflexion

sont largement abordées dans [Robein E., 1999]. Elles font intervenir différentes étapes dont, en premier lieu, la *sommation* (*"stack"* en anglais) dans le domaine de l'*offset nul*. En effet, la *couverture multiple* en sismique réflexion consiste à "éclairer" un même point du sous-sol par des couples source-capteur différents. Ainsi, par sommation, le rapport signal/bruit au niveau de ce point est amélioré. Cependant, cette sommation nécessite au préalable une correction verticale de la position du point sur chaque trace, liée à l'angle d'incidence avec lequel il a été éclairé au niveau de chaque couple source-capteur. Cette transformation dans le domaine de l'offset nul est réalisée par la *correction d'obliquité* ou *NMO* (de l'anglais "Normal Move Out") et la *correction d'obliquité tenant compte du pendage* ou *DMO* (de l'anglais "Deep Move Out"). Ces deux méthodes font intervenir la phase semi-automatique et fastidieuse de *pointé des vitesses*. Outre les corrections verticales avant sommation, le traitement sismique fait intervenir après sommation des corrections horizontales qui consistent à déplacer un point vers sa position horizontale réelle en profondeur, une étape appelée *migration*. Toutes ces étapes extrêmement complexes, et notamment la phase de migration, doivent être effectuées en 3D sous peine de commettre des erreurs inhérentes au raisonnement en 2D [Brown A.R., 2004].

À ce stade, du fait de l'enregistrement en temps du signal, toute coordonnée correspondant à un événement observé sur une trace sismique doit être interprétée comme un temps de parcours source-réflecteur-capteur (surface-réflecteur-surface dans le domaine de l'offset nul) de l'onde acoustique dans le sol et non comme une profondeur de réflecteur. Ainsi, au cube sismique est associé un *modèle de vitesse* qui permet d'effectuer une *conversion temps-profondeur* afin de transformer l'axe vertical en temps du cube en un axe en profondeur indispensable à une interprétation tridimensionnelle intuitive des structures.

Le résultat de ce traitement est une véritable image tridimensionnel du sous-sol : l'image sismique. Elle est en fait représentée sous la forme d'une grille régulière dans laquelle chaque *voxel*[6] stocke une valeur d'amplitude sismique (Figure 1.4-B). Dans ce cas, les axes correspondent effectivement à des coordonnées spatiales et l'image est donc directement interprétable par un géophysicien et/ou un géologue.

1.2 L'interprétation sismique dans la chaîne d'exploration-production

Dans les phases qui précèdent la mise en production à proprement parler d'un champ, une première phase exploratoire consiste à mettre en évidence les éventuels signes de la présence d'hydrocarbures dans un bassin sédimentaire. À cette première approche à dominante qualitative succède une phase de modélisation dont l'objectif est de statuer sur la viabilité économique d'un gisement une fois mis en production. La chaîne de modélisation que nous allons présenter maintenant couvre les différentes étapes intervenant dans cette deuxième phase exploratoire. Nous verrons ensuite quels sont les différents apports de la sismique dans l'ensemble de la phase exploratoire et comment s'organise les différentes étapes de l'interprétation sismique.

1.2.1 La chaîne de modélisation en exploration-production

Après une première approche qualitative du bassin sédimentaire basée sur différentes formes de données (profils sismiques, données de puits, relevés de terrain, ...) dans laquelle les grandes structures géologiques ont pu être mises en évidence et le contexte sédimentaire établi, le géo-

[6] contraction de l'anglais "volume element" (élément de volume), par analogie avec pixel, contraction de "picture element" (élément d'image) ; voir Section 2.1

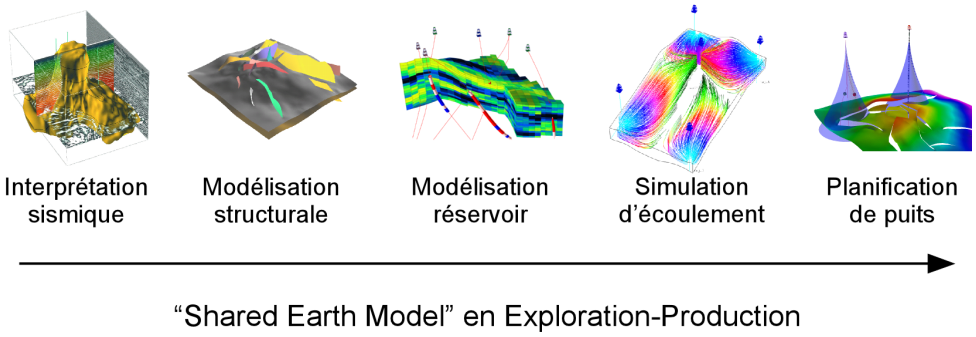

FIG. 1.5 – Les différentes étapes intervenant dans la chaîne de modélisation en exploration-production.

physicien et/ou le géologue sont en mesure de déterminer quelques zones clés qui représentent de potentiels gisements d'hydrocarbures. Dès lors, une phase de modélisation quantitative peut être menée dont l'objectif est de chiffrer la viabilité des gisements ainsi mis en évidence. Cette seconde phase intervient à plus petite échelle, celle du réservoir, tandis que la première phase abordait le bassin sédimentaire dans son ensemble. Elle repose sur la construction d'un modèle numérique de réservoir enrichi au cours d'étapes successives faisant intervenir des spécialistes issus de tous les domaines de l'exploration-production : géophysiciens, géologues, géostatisticiens, ingénieurs réservoir, foreurs... Ce modèle ainsi partagé et riche d'expériences diverses est communément appelé en anglais le *"Shared Earth Model"*. Nous allons nous intéresser aux étapes majeures de sa construction (Figure 1.5).

Interprétation sismique

L'image sismique, dont la génération a été largement abordée au cours de la Section 1.1, constitue l'élément essentiel de cette première étape. À partir de cette image tridimensionnelle voxelisée, l'objectif premier est de mettre en évidence les principales interfaces géologiques [BROWN A.R., 2004]. Pour simplifier, celles-ci sont de deux sortes : les *horizons*, sismiques ou géologiques suivant le contexte, constituent les limites entre deux couches aux propriétés pétrophysiques (et donc acoustiques) différentes, tandis que les *failles* sont des discontinuités plus ou moins verticales affectant les couches géologiques et résultant de l'histoire géomécanique du bassin. Les horizons sont clairement marqués par des contrastes d'impédance acoustique le long du signal sismique, au contraire des failles qui apparaissent au travers des décalages qu'elles engendrent au sein des horizons.

Modélisation structurale

Les interfaces mises en évidence lors de la première étape d'interprétation sismique sont extraites de l'image voxelisée sous forme surfacique [DORN G.A., 1998]. Ainsi, d'une image tridimensionnelle, est extrait un modèle surfacique du sous-sol qui fait la synthèse des interfaces géologiques ayant un intérêt dans l'évaluation du gisement : le *modèle structural*. Il s'agit ensuite d'étudier les relations chronologiques et d'interdépendance qui existent entre les différentes surfaces du modèle ; en définissant par exemple l'ensemble des horizons affectés par une faille donnée ou encore l'ordre chronologique dans lequel une série de failles a affecté un même horizon. Ces relations sont exprimées sous la forme de contraintes géométriques attachées aux objets du modèle [MALLET J.L., 1989].

Modélisation réservoir

Le modèle structural ainsi obtenu contient uniquement les informations sur la géométrie des frontières entre les couches géologiques. La modélisation réservoir a pour but de passer d'une représentation surfacique à une représentation volumique dans laquelle les compartiments isolés par les interfaces du modèle structural sont discrétisés [Mallet J.L., 2002] et forment le support de simulations géostatistiques [Goovaerts P., 1997] dont le but est de représenter l'évolution spatiale des propriétés pétrophysiques du sous-sol.

Simulation d'écoulement

À partir du modèle volumique de réservoir, des calculs d'écoulement permettent, en fonction de différents scénarios de puits injecteurs-producteurs, d'estimer les volumes d'hydrocarbures récupérables et de statuer sur la viabilité économique d'un gisement [Aziz K. et Settari A., 1979].

Planification de puits

En dernier lieu, la planification des puits se base sur le scénario de production le plus adéquat en fonction des contraintes de coût et de faisabilité.

L'interprétation de l'image sismique intervient de manière évidente dans la première étape de cette chaîne de modélisation. Il serait cependant réducteur d'en résumer la portée à la seule construction du modèle structural. Avant de discuter des principaux recours à l'information sismique dans l'ensemble de la phase exploratoire, nous allons présenter les différentes questions auxquelles l'image sismique est en mesure d'apporter des réponses.

1.2.2 Les différentes formes d'analyse de l'image sismique

L'image sismique met en évidence les contrastes relatifs d'impédance acoustique dans le sous-sol (Section 1.1.2). Outre la source d'information structurale évidente qu'elles apportent et bien que ne pouvant être interprétées de manière absolue, les amplitudes sismiques fournissent également des informations d'ordre pétrophysique qui renseignent sur la nature des roches dans les formations géologiques ainsi que sur la nature des contacts. L'analyse de l'image sismique prend ainsi des formes variées selon le type d'information recherché.

Analyse stratigraphique

L'analyse stratigraphique de l'image sismique concerne en premier lieu l'analyse de *séquences sismiques* [Selley R.C., 1998], une séquence sismique étant un bloc de dépôts successifs appartenant à un même environnement de dépôt. Ce sont typiquement des couches plus ou moins parallèles sur l'image sismique. La corrélation à échelle régionale des séquences sismiques constitue la *stratigraphie séquentielle* [Vail P.R. *et al.*, 1977]. Dans un deuxième temps, l'observation des relations entre réflecteurs aux frontières de séquences permet de caractériser la succession des environnements de dépôt [Sheriff R.E., 1976]. Cette deuxième phase de l'analyse stratigraphique est appelée analyse de *faciès sismiques* [Selley R.C., 1998].

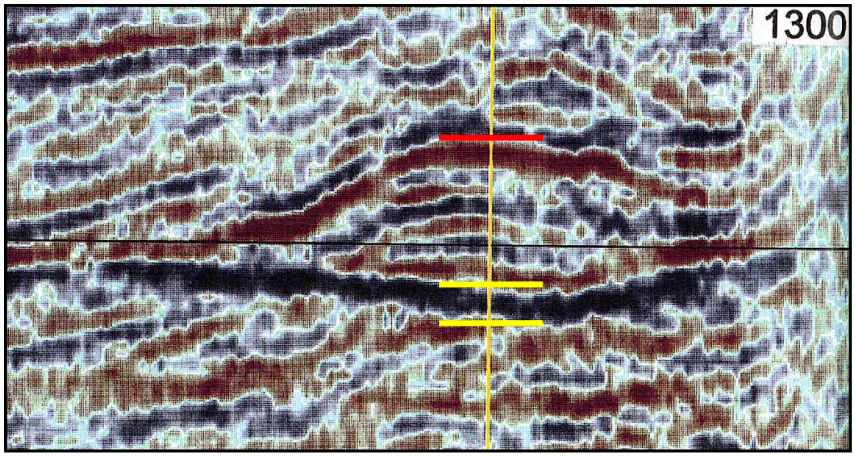

FIG. 1.6 – Anomalie positive de l'amplitude sismique, ou *"bright spot"*, correspondant à un contact eau-gaz dans un réservoir offshore du delta du Nil (Égypte). Les marqueurs jaunes encadrent le "bright spot" et le marqueur rouge correspond au toit du gaz (d'après [BROWN A.R., 2004]).

Analyse structurale

L'analyse structurale est la plus intuitive des formes d'analyse de l'image sismique. En effet, les amplitudes sismiques marquant des contrastes d'impédance acoustique, donc de propriétés pétrophysiques, les principaux réflecteurs sont la signature de discontinuités géologiques majeures. Ces interfaces entre couches géologiques de nature différente sont appelées horizons géologiques tandis que, dans le contexte de l'image sismique, le terme d'horizon sismique fait directement référence aux réflecteurs qui en sont l'expression acoustique. De même, les discontinuités latérales observées au niveau des réflecteurs sont la signature de failles ayant affecté les couches géologiques au cours de l'histoire du bassin. Notons également du domaine de l'interprétation structurale, la mise en évidence des dômes de sel qui dispersent les ondes sismiques et rendent difficile l'imagerie des couches sous-jacentes [O'BRIEN M.J. et GRAY S.H., 1996].

Analyse pétrophysique

L'analyse pétrophysique s'intéresse aux propriétés du signal sismique (polarité, amplitude, fréquence, ...) pour tirer des conclusions sur la nature exacte des roches en place. L'analyse d'*attributs sismiques* [TANER M.T. et SHERIFF R.E., 1977], qui n'est pas exclusivement pétrophysique, intervient pour une large part dans cette forme d'interprétation de la sismique. L'un des éléments importants de l'analyse pétrophysique est la recherche d'*indicateurs d'hydrocarbures directs*, tels que les *flat spots* [BACKUS M.M. et CHEN R.L., 1975]. Un flat spot se définit par l'existence d'un réflecteur horizontal qui traverse un ensemble de réflecteurs plongeant parallèlement. Ils sont généralement les témoins de contacts eau-hydrocarbures et présentent donc une importance particulière. Le fort contraste d'impédance acoustique entre la roche inférieure contenant de l'eau et la roche supérieure contenant des hydrocarbures (pétrole ou gaz) se traduit par une forte anomalie de l'amplitude sismique appelée *"bright spot"* (Figure 1.6).

Ainsi l'interprétation sismique ne se résume pas à la détermination de la structure du réservoir, mais répond également à des questions sur la qualité des réservoirs et des pièges, sur la maturation et la migration des hydrocarbures... Nous allons maintenant voir comment ces

différentes formes d'analyse de l'image sismique interviennent dans les phases exploratoires successives d'un bassin sédimentaire.

1.2.3 La démarche interprétative

Nous avons vu qu'il y avait schématiquement deux phases exploratoires dans l'étude d'un bassin, l'une qualitative à l'échelle régionale, l'autre quantitative à l'échelle du réservoir. Du point de vue de l'interprétation sismique, il est possible de distinguer parallèlement deux phases d'interprétation suivant l'échelle considérée : la macro-interprétation et la micro-interprétation [Christie M., 2002].

Macro-Interprétation

La macro-interprétation constitue l'interprétation des données sismiques à échelle régionale, celle du bassin sédimentaire dans son ensemble. Elle fait intervenir préférentiellement l'analyse stratigraphique afin de déterminer précocement les environnements de dépôt successifs que le bassin a connus au cours de son histoire. Le second point essentiel de cette approche régionale est l'analyse structurale à grande échelle qui fournit une vue globale des contraintes géomécaniques qu'a pu subir le bassin au cours de son histoire. L'analyse pétrophysique occupe à ce stade une place marginale, intervenant ponctuellement de manière qualitative.

Micro-Interprétation

Les réservoirs potentiels ayant été mis en évidence par les environnements de dépôt et la structure à grande échelle, il est nécessaire de modéliser quantitativement leur structure à plus petite échelle pour repérer les éventuels pièges à hydrocarbures. Dans cette phase, l'analyse structurale prend une importance primordiale. Elle est la base de la chaîne de modélisation présentée à la Section 1.2.1. Par ailleurs, les informations issues d'une analyse pétrophysique avancée sont très importantes pour caractériser la nature des roches en présence lors de l'étape de modélisation réservoir de cette même chaîne.

Fort d'une connaissance plus approfondie des besoins de l'interprétation sismique, il nous sera maintenant possible de mettre en œuvre des outils adéquats pour visualiser les données en question. Nous allons tout d'abord dresser un aperçu des outils de visualisation disponibles de manière standard à l'heure actuelle et de leur historique.

1.3 Visualisation des données sismiques

Jusque dans les années 1970, les techniques d'imagerie sismique ainsi que les outils de visualisation des données sismiques n'ont subi que peu d'évolutions. Les données étaient acquises en deux dimensions le long de simples profils verticaux qui étaient imprimés et interprétés directement sur papier. À partir de la fin des années 1970 et du début des années 1980, la sismique a connu deux révolutions majeures intimement liées aux progrès de l'informatique. La première révolution concerne l'imagerie qui, grâce à la puissance de calcul fournie par l'outil informatique, a pu étendre les techniques d'acquisition et surtout de traitement des profils sismiques 2D à de véritables campagnes 3D. La seconde révolution concerne la visualisation qui, avec l'apparition

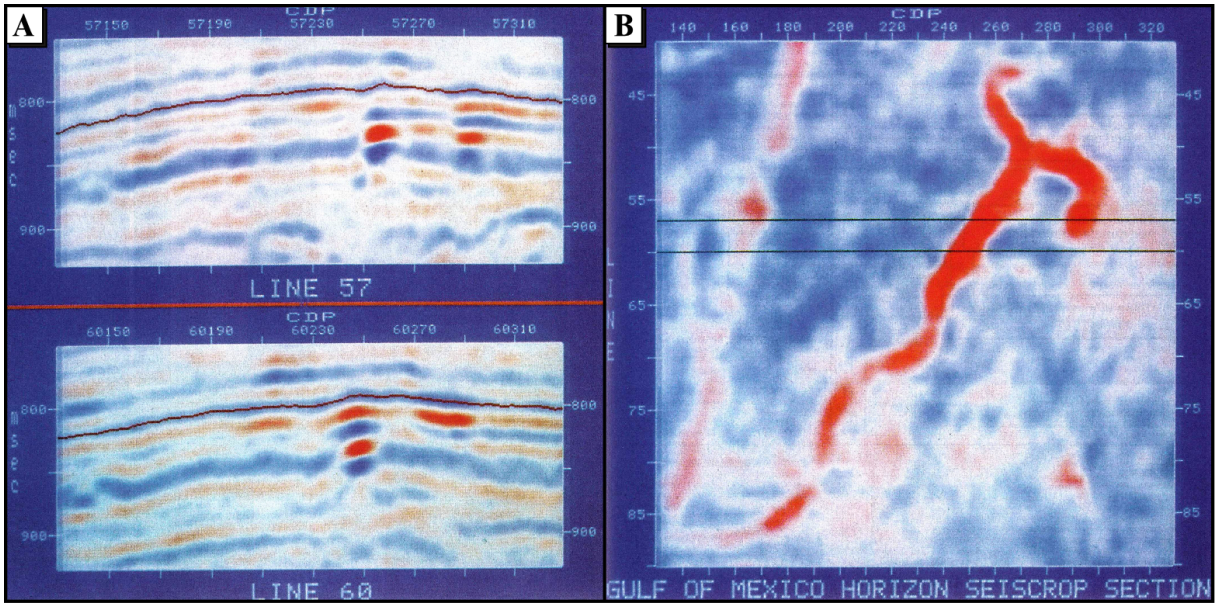

FIG. 1.7 – Mise en évidence de chenaux à l'aide d'une *coupe horizon* dans une image sismique du Golfe du Mexique. (A) Deux inlines montrent la trace d'un horizon extrait (marron) surmontant des événements sismiques intéressants (rouge). (B) Une *coupe horizon* calée sur l'horizon marron et extraite à travers ces événements met en évidence des chenaux. Les lignes noires horizontales sur la coupe horizon indiquent la position des inlines (d'après [BROWN A.R., 2004]).

des stations d'interprétation, a vu le support papier classique progressivement remplacé par un support digital. Ces stations d'interprétation ne se contentaient pas de fournir de simples versions digitales des précédents profils papier. En effet, elles permettaient d'effectuer des coupes dans tous les axes du cube sismique : *inlines* et *crosslines* pour les profils verticaux classiques, et *coupes temps* ou *coupes profondeur*, selon que le modèle de vitesse a été appliqué ou non (Section 1.1.3), pour les coupes dans l'axe vertical du cube. Un autre type de coupes largement utilisé est la *coupe horizon* qui consiste, plutôt qu'à visualiser les amplitudes sismiques sur des coupes horizontales à temps ou profondeur constants dans le cube, à extraire les amplitudes à une distance fixe d'un horizon sismique repéré sur les profils verticaux. Comme l'atteste la Figure 1.7, cette approche s'avère particulièrement intéressante pour mettre en évidence la présence de chenaux au voisinage d'horizons par exemple.

Les apports de la sismique 3D pour la mise en évidence de gisements d'hydrocarbures sont considérables et se sont faits réellement ressentir dans les années 1990. C'est à cette période que les grandes compagnies ont engagé une véritable politique de campagnes 3D. Ainsi, vers la fin des années 1990, la compagnie Amoco qui cherchait à évaluer l'impact de la sismique 3D sur ses résultats en est venue à la conclusion que le taux de succès qu'ils enregistraient pour le forage de puits était passé de 14% en 1990 à 47% en 1997, tandis que la proportion de champs couverts par la 3D était passée de 5% à 97% pendant la même période [BROWN A.R., 2004].

Cependant, aujourd'hui encore la majeure partie de l'interprétation de ces campagnes 3D repose sur l'utilisation exclusive d'outils 2D, à savoir les quatre types de coupes cités précédemment (inlines, crosslines, coupes temps/profondeur et coupes horizon). L'apparition des stations graphiques au cours des années 1990 s'est accompagnée d'outils de visualisation des données sismiques permettant de replacer ces coupes 2D dans un contexte 3D et de les manipuler en temps réel. Ces nouveaux outils sont également désormais capables de visualiser les coupes horizon en

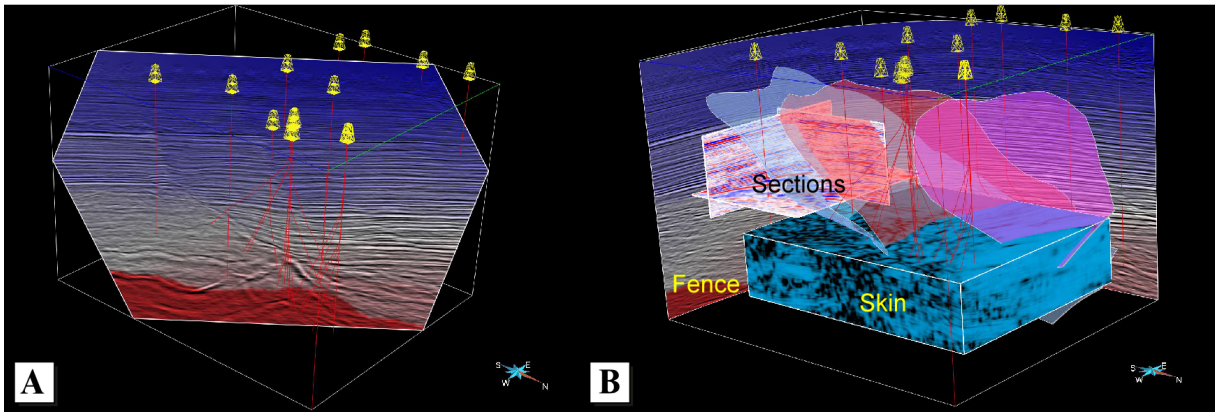

FIG. 1.8 – Principales sondes du logiciel VolumeExplorer [BOSQUET F. et DULAC J.C., 2000]. (A) Le *Slicer* effectue une coupe arbitraire dans l'ensemble du volume. (B) La *Fence* permet de visualiser des panneaux verticaux, les *Sections* des coupes orthogonales et la *Skin* la frontière d'un sous-volume.

perspective, plutôt qu'aplanies comme c'est le cas avec les stations 2D classiques, mais aussi et surtout d'extraire des coupes complètement arbitraires. Ainsi, le module de visualisation de volumes sismiques VolumeExplorer du logiciel Gocad sur lequel nous nous sommes basés pour ces travaux définit un certain nombre de sondes ("probes" en anglais) qui permettent d'extraire les amplitudes sismiques selon des techniques différentes [BOSQUET F. et DULAC J.C., 2000]. La Figure 1.8 montre les principales. Le *Slicer* est une coupe arbitraire dans l'ensemble du volume qui présente un intérêt particulier en macro-interprétation, notamment pour l'analyse structurale. La *Fence* est une coupe qui définit un certain nombre de panneaux verticaux. En faisant coïncider ces panneaux avec des puits d'exploration, ce type de sondes facilite les corrélations lors de l'analyse stratigraphique. Les *Sections* sont de simples coupes d'extension arbitraire orthogonales aux axes du cube. Finalement, la *Skin* représente la frontière d'un sous-volume arbitraire qui permet d'explorer localement les données. Relativement récents, ces outils trouvent progressivement une population d'utilisateurs désireux d'intégrer de manière plus intuitive le raisonnement 3D dans l'interprétation. La valeur ajoutée est considérable, réduisant parfois les temps d'interprétation de plusieurs semaines à quelques jours, voire quelques heures [CHOPRA S. *et al.*, 2005].

Il est intéressant de noter que, bien que placés dans un contexte 3D, les outils standard de visualisation des données sismiques restent des outils 2D. En effet, l'apparition des techniques de visualisation volumique sur stations graphiques au début des années 1990, impulsée par le domaine de l'imagerie médicale où elle constitue une véritable révolution, n'a eu que peu d'échos dans le monde de la sismique. Ces techniques, que nous présenterons en détails au Chapitre 2, permettent de visualiser le volume en transparence et ainsi d'avoir de réelles informations sur sa structure dans les trois dimensions. L'une des principales raisons pour lesquelles l'interprétation sismique tarde à adopter ces méthodes, sur laquelle nous aurons l'occasion de revenir au Chapitre 3, est l'inadaptation des techniques de visualisation volumique classiques aux contraintes des données sismiques (forte fréquence spatiale le long de la dimension verticale, bruit, ...). Or, au début des années 2000 la puissance graphique des PCs a connu, sous l'impulsion de l'industrie du jeu vidéo, une véritable explosion et dépasse aujourd'hui de loin celle des stations graphiques pour un prix bien inférieur. Ceci a ouvert de nouvelles voies, notamment dans l'amélioration des techniques de visualisation volumique, aussi bien du point de vue de la qualité des images que de la performance. Ces évolutions tardent à produire leurs effets dans le domaine de la visualisation en géosciences d'une manière générale et nous verrons comment l'interprétation sismique

en particulier peut en bénéficier.

Le présent chapitre visait à replacer notre travail dans son contexte applicatif, à savoir l'interprétation des données sismiques dans l'exploration-production des hydrocarbures. Nous allons maintenant nous intéresser au cadre scientifique dans lequel s'inscrit notre travail, à savoir celui de la visualisation de données volumiques.

Chapitre 2

Contexte scientifique : les données volumiques et leur visualisation

Au cours de ce chapitre nous allons faire abstraction du domaine d'application qu'est l'interprétation sismique pour dresser un tableau récapitulatif des techniques de visualisation volumique. Celle-ci ayant fait l'objet de recherches particulièrement intensives au cours des quinze dernières années, notre but est de présenter les grandes tendances et leur évolution afin de saisir les enjeux du problème. Il est préalablement important de présenter les notions de base attachées aux données volumiques. Nous verrons ensuite les principales techniques de visualisation volumique classées en méthodes directes et indirectes, et leur mise en œuvre. L'une d'entre elles, le rendu volumique, particulièrement importante dans le cadre de ce mémoire, sera présentée plus en détail à travers les implantations qui ont émergé de l'évolution du matériel graphique. Finalement, nous verrons comment des méthodes existantes fondées sur l'utilisation de plusieurs processeurs en parallèle connectés par un réseau rapide permettent de visualiser de plus grandes quantités de données.

2.1 Les données volumiques

Les données volumiques ont des origines et des significations variées suivant le domaine d'application concerné. Elles sont en effet obtenues aussi bien par simulation (écoulement de fluide, évolution d'un plasma au cours du temps, ...) que par mesure (imagerie par résonance magnétique, tomographie, ...). Dans le cadre de notre étude, nous avons vu au chapitre précédent qu'il s'agit de mesures d'amplitudes sismiques. Il existe différents types de représentation de ces données [Foley J.D. *et al.*, 1996]. Dans tous les cas, il s'agit de représenter un champ scalaire S défini dans un volume Ω de l'espace. Nous allons nous restreindre ici au cas des *représentations discrètes*, également appelées *représentations cellulaires*, *grilles* ou encore *maillages* volumiques. Dans ce mode de représentation, un ensemble de N valeurs discrètes s_i du champ scalaire est connu en des points $\mathbf{x}_i$ de l'espace :

$$S(\mathbf{x}_i) \;=\; s_i \;,\; i = 1 \ldots N \;\mid\; \mathbf{x}_i \;\in\; \mathbb{R}^3 \text{ et } s_i \;\in\; \mathbb{R} \tag{2.1}$$

Les points d'échantillonnage $\mathbf{x}_i$ constituent les nœuds de la grille. Ils sont reliés par des arêtes qui elles-même forment des cellules. Les relations de voisinage ainsi définies entre les nœuds constituent la *topologie* de la grille. Il existe différentes familles de grilles selon le schéma

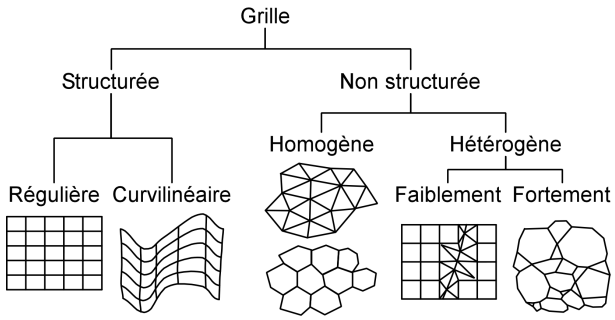

FIG. 2.1 – Taxinomie des représentations cellulaires en fonction de leur *topologie* (d'après [CAUMON G. *et al.*, 2005]).

d'organisation des nœuds dans l'espace (Figure 2.1). La première distinction porte sur le nombre de voisins dont dispose chacun des nœuds de la grille. Dans le cas où ce nombre est variable, la topologie est dite *explicite* et la grille *non structurée*. Les cellules sont des polyèdres arbitraires, le plus souvent des tétraèdres. Lorsque différents types de polyèdres sont présents, la grille est dite *hétérogène*, *homogène* dans le cas contraire. Lorsque le nombre de voisins par nœud est constant, la topologie est dite *implicite* et la grille *structurée*. Les cellules sont généralement des hexaèdres, soit réguliers, la grille est alors dite *régulière*, soit irréguliers, elle est dite *structurée irrégulière* ou *curvilinéaire*. Dans le cas précis des données sismiques, la disposition géographique des capteurs de surface à intervalles réguliers de même que l'échantillonnage vertical à pas de temps constant (Section 1.1.3) autorisent le stockage sous forme de grilles régulières. Dans le cadre de la modélisation en exploration-production, l'utilisation des grilles irrégulières, structurées ou non, intervient essentiellement dans les étapes de modélisation réservoir et de simulation d'écoulement (Section 1.2.1). Pour cette raison, nous allons nous concentrer sur les techniques de visualisation volumique dans les grilles régulières, dont les concepts généraux s'étendent aux autres types de grilles.

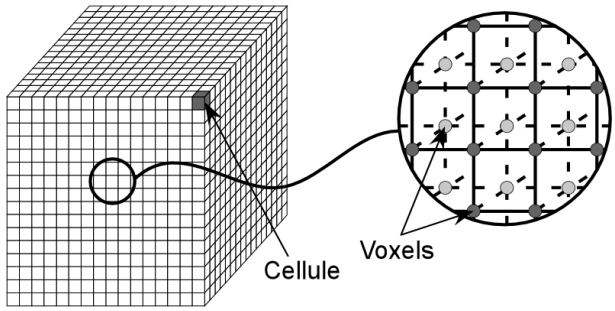

FIG. 2.2 – Distinction cellule-voxel dans une représentation cellulaire régulière. L'agrandissement montre deux couches de voxels (superficielle en gris foncé et au second plan en gris clair) correspondant aux points d'échantillonnage. La cellule représentée en gris foncé sur la vue d'ensemble est délimitée par huit voxels (sept superficiels et un de second plan).

La définition d'une grille régulière est intimement liée à la notion de *voxel*, extension du pixel au cas des volumes. De manière courante, il est défini comme l'élément de volume unitaire et de valeur constante autour d'un point d'échantillonnage [KAUFMAN A., 1994]. Bien qu'intuitive, cette définition est incorrecte et source de confusions lorsqu'il s'agit d'appliquer un filtre de reconstruction sur un ensemble de voxels [SMITH A.R., 1995]. Dans un souci de cohérence

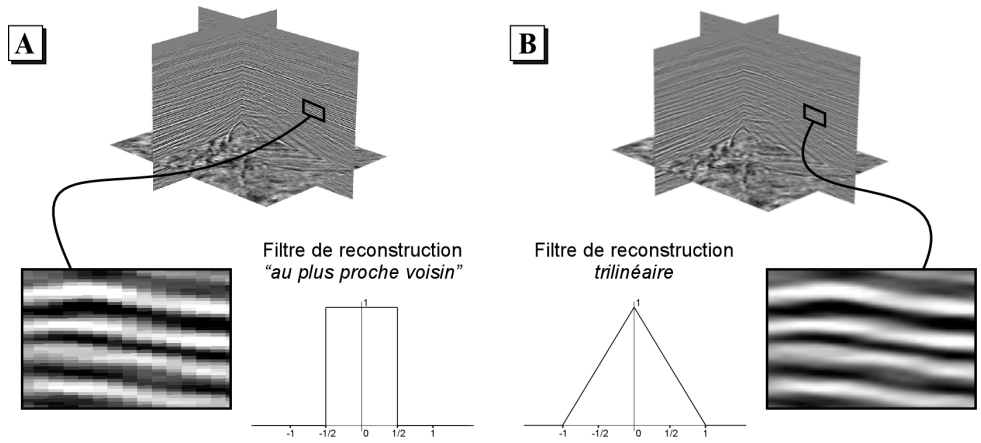

FIG. 2.3 – Résultats comparés du filtrage de l'image sismique par *interpolation "au plus proche voisin"* (A) et par *interpolation trilinéaire* (B).

vis-à-vis de la théorie du traitement du signal et comme le montre la Figure 2.2, il est plus approprié de confondre le voxel avec le point d'échantillonnage [LICHTENBELT B. *et al.*, 1998, REZK-SALAMA C. *et al.*, 2004, PFISTER H., 2005].

Le champ scalaire S peut être considéré comme un signal continu $S(\mathbf{x})$ dans l'espace. La grille régulière constitue le support de discrétisation de ce signal. Ainsi, en tout point $\mathbf{x}$ de l'espace, il est possible de reconstruire une approximation $s(\mathbf{x})$ du signal continu par convolution des valeurs s_i aux nœuds de la grille avec un filtre de reconstruction centré en $\mathbf{x}$ [MÖLLER T. *et al.*, 1999]. Bien que plus précis, les filtres d'ordre supérieur à 1 sont, à quelques exceptions près [HADWIGER M. *et al.*, 2001], prohibés car trop coûteux. En effet, dans la pratique, les filtres les plus couramment utilisés sont l'*interpolation "au plus proche voisin"* et l'*interpolation linéaire*, ou plutôt l'*interpolation trilinéaire* dans le cas présent. La Figure 2.3 donne la représentation 1D de ces filtres ainsi que le résultat de l'interpolation sur une section de cube sismique. L'interpolation trilinéaire fournit le meilleur rapport entre qualité de rendu et performance. D'autre part, ce filtre est implanté de manière très efficace par le matériel graphique. Dans chaque cellule, un interpolant de la forme :

$$s(\mathbf{x}) \;=\; s(x,y,z) \;=\; a \;+\; bx \;+\; cy \;+\; dz \;+\; exy \;+\; fxz \;+\; gyz \;+\; hxyz \qquad (2.2)$$

est défini à partir des huit valeurs aux nœuds. Lorsque, dans la suite de ce mémoire, nous ferons référence à l'interpolation sans plus de précisions, nous supposerons que le signal $s(\mathbf{x})$ est reconstruit par un filtre trilinéaire.

Indépendamment du domaine d'application scientifique concerné, la finalité de la visualisation volumique est de fournir à l'observateur une image porteuse d'information sur la structure du champ scalaire sous-jacent. Ni la beauté de l'image produite au sens esthétique du terme ni même son réalisme ne sont les objectifs suivis. Il est important de préciser ce dernier point car la notion de réalisme dans notre cas ne peut s'avérer objective. En effet, nous cherchons à "imager" un phénomène qui, par définition, est invisible, ce qui signifie que toute ressemblance avec des effets visuels réalistes n'a pour autre but que de placer l'observateur dans un cadre d'interprétation qui lui soit familier. D'autre part, l'illustration de techniques à l'aide d'objets volumiques de la vie courante permet d'évaluer la pertinence des images qu'elles produisent de manière objective.

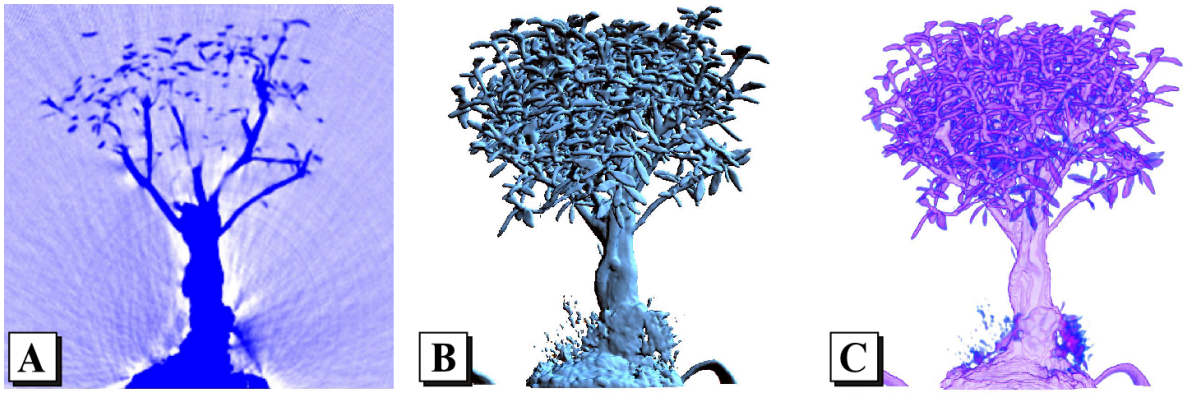

FIG. 2.4 – Visualisation de données de scanner d'un bonsaï par *section* (A), *surface d'isovaleur* (B) et *rendu volumique* (C).

2.2 Les techniques de visualisation volumique

Différentes techniques permettent de visualiser un champ scalaire volumique. Il est possible de les classer en deux grandes familles : les *méthodes indirectes* et les *méthodes directes* [REZK-SALAMA C., 2001]. La distinction porte sur l'extraction préalable ou non d'un sous-ensemble des données brutes pour le visualiser indépendamment de celles-ci.

2.2.1 Méthodes indirectes

Les méthodes indirectes portent également le nom de *méthodes d'extraction*. Il est ainsi fait référence à l'ensemble des techniques de visualisation d'un champ scalaire par extraction préalable d'un sous-ensemble des données brutes. Ce sous-ensemble, généralement extrait sous forme surfacique et représenté par un ensemble de polygones, est alors visualisé indépendamment des données volumiques à l'aide des techniques de rendu polygonal conventionnelles (Section 2.3.1). Ces méthodes supposent donc une étape de prétraitement des données pour en extraire l'information recherchée.

Sections

La *visualisation par section* consiste à extraire un plan orthogonal à un axe du volume et à visualiser le champ scalaire sur ce plan à l'aide d'un code de couleur (Figure 2.4-A). La classification de cette méthode dans les méthodes indirectes est sujette à discussion selon le mode d'implantation de la technique. Cependant, et bien que fort utile pour l'interprétation (Section 1.3), la visualisation par section ne présente pas de difficultés algorithmiques particulières. En revanche, d'un point de vue purement technique, l'implantation pour des volumes de données de très grande taille, de l'ordre de la centaine de gigaoctets, peut devenir complexe à mettre en œuvre ; nous y reviendrons en conclusion du Chapitre 6.

Surfaces d'isovaleur

L'une des principales techniques de visualisation par méthode indirecte est la *visualisation par surface d'isovaleur* (Figure 2.4-B). Une surface d'isovaleur est une surface équipotentielle

du champ scalaire, également appelée *isosurface*. Elle est définie, pour une *isovaleur* scalaire k, comme le sous-ensemble Ω_k de Ω vérifiant :

$$\Omega_k \;=\; \{\; \mathbf{x} \mid s(\mathbf{x}) = k \;\} \tag{2.3}$$

Une surface d'isovaleur fournit ainsi une interprétation géométrique du champ scalaire. La compréhension de sa structure tridimensionnelle est facilitée par l'utilisation de méthodes d'illumination dépendantes de la normale en chacun des points de la surface et de sources lumineuses artificielles [Foley J.D. *et al.*, 1996]. Ceci est facilité par l'utilisation de matériel graphique standard (Section 2.3.1).

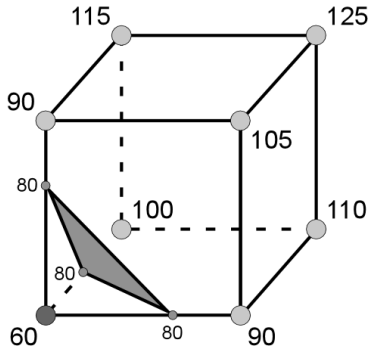

Fig. 2.5 – Triangulation de l'intersection d'une surface d'isovaleur ($k = 80$) avec une cellule hexaédrique dans l'algorithme des *"Marching Cubes"*. Les voxels positifs et négatifs sont respectivement représentés en gris clair et gris foncé. Les points d'intersection sur les arêtes sont obtenus par interpolation linéaire des valeurs aux voxels.

La principale difficulté des méthodes d'extraction de surfaces d'isovaleur réside dans la conversion des données volumiques brutes en une représentation surfacique à base de polygones exploitable par le matériel graphique. Dans le cas particulier des grilles régulières, ce prétraitement a fait l'objet de nombreuses recherches dont la plupart se fondent sur l'algorithme dit des *"Marching Cubes"* [Lorensen L. et Cline H., 1987]. Chaque cellule hexaédrique formée par huit voxels voisins est traitée de manière indépendante (Figure 2.5). Selon que la valeur du champ scalaire en chacun des voxels est supérieure ou inférieure à l'isovaleur k de la surface à extraire, ce voxel est marqué comme positif ou négatif. Sur les arêtes reliant deux voxels de signe contraire, le point d'intersection avec la surface est calculé par interpolation linéaire de la valeur aux extrémités. Finalement, l'intersection de la surface avec la cellule courante est construite par triangulation à partir des points d'intersection calculés sur les arêtes. Il existe $2^8 = 256$ configurations possibles pour chaque cellule. Par symétrie, ces 256 configurations sont ramenées à 15 configurations types. Lorensen et Cline classent ces configurations dans une table qui permet un accès rapide à la triangulation en fonction de l'état des voxels. Une des limites majeures des "Marching Cubes" est la possibilité de voir apparaître des ambiguïtés qui conduisent dans certains cas à des incohérences topologiques [Durst M., 1988]. Le *décideur asymptotique* (*"asymptotic decider"* en anglais) [Nielson G.M. et Hamann B., 1991] est un moyen de lever ces ambiguïtés. L'autre limitation importante concerne le grand nombre de polygones générés lors de ce prétraitement, celui-ci pouvant être réduit par fusion des arêtes et faces non significatives [Moore D. et Warren J., 1991] ou par les méthodes classiques de décimation de surface [Schroeder W. *et al.*, 1992, Hoppe H., 1996]. D'autres techniques de triangulation de la surface à l'intérieur de chaque cellule intersectée ont été proposées, notamment en tournant autour des faces de la cellule par le recours à des liens topologiques explicites entre les nœuds, faces et

arêtes [Bloomenthal J., 1988, Lévy B. *et al.*, 2001, Caumon G. *et al.*, 2005]. Des approches plus récentes exploitent les capacités du matériel graphique moderne (Section 2.3.1) pour lui faire directement calculer cette triangulation [Pascucci V., 2004, Kipfer P. et Westerman R., 2005, Buatois L. *et al.*, 2006].

Il est important de préciser que, pour une grille contenant n cellules, le nombre moyen de cellules intersectées par une surface d'isovaleur est $O(n^{2/3})$ [Itoh T. et Koyamada K., 1995]. Par conséquent, un algorithme naïf qui parcourt toutes les cellules de la grille passe la majorité de son temps à tester des cellules non intersectées. Ceci peut être évité en regroupant les cellules de telle sorte qu'elles puissent être disqualifiées en blocs. Il est ainsi possible de les regrouper dans l'espace des valeurs du champ scalaire en utilisant un *arbre d'intervalles* (*"interval tree"* en anglais) [McCreight E.M., 1985], ou dans l'espace géométrique par subdivision du volume à l'aide d'un *arbre octal* (*"octree"* en anglais) [Wilhelms J. et van Gelder A., 1992]. Une approche alternative dite des *"contour seeds"* [Bajaj C.L. *et al.*, 1996] consiste à parcourir la grille à partir de cellules graines.

Malgré toutes ces techniques d'optimisation, les méthodes indirectes par extraction de surfaces d'isovaleur continuent de souffrir de temps de prétraitement parfois prohibitifs. Ainsi, dans certains cas, la visualisation interactive avec modification en temps réel de la valeur k associée à l'isosurface est impossible. D'autre part, l'extraction d'une surface, même plongée dans un espace 3D, conduit inéluctablement à une perte d'information sur le champ scalaire observé. Ceci se vérifie d'autant plus qu'il est difficile de s'affranchir des effets d'occlusion, les structures internes étant souvent masquées par les faces externes. Un moyen de dépasser ces limitations inhérentes aux méthodes indirectes est de recourir aux méthodes directes.

2.2.2 Méthodes directes

Par opposition aux méthodes indirectes, les méthodes directes visualisent les données volumiques brutes dans leur ensemble sans procéder à l'extraction préalable d'un sous-ensemble. Ces méthodes, auxquelles il est communément fait référence sous le terme générique de *rendu volumique*, ont vu le jour au milieu des années 1980 [Kajiya J.T. et von Herzen B.P., 1984, Debrin R. *et al.*, 1988, Sabella P., 1988].

Intégrale de rendu volumique

L'approche classique considère le volume Ω comme un milieu semi-transparent chargé de particules élémentaires (Figure 2.4-C). Ces particules possèdent des propriétés optiques qui permettent de simuler la propagation de la lumière dans le milieu à partir de modèles de comportement optique plus ou moins évolués [Max N., 1995]. Ces modèles décrivent le comportement de chaque particule vis-à-vis de son environnement. Le plus courant est le *modèle d'émission-absorption* dans lequel chacune des particules du volume Ω émet un rayonnement lumineux dans son environnement et absorbe une partie du rayonnement incident, le reste étant transmis sans modification. À partir de ce modèle, il est possible d'accumuler la contribution de chaque particule du volume le long d'un rayon lumineux virtuel issu de l'œil d'un observateur et de mesurer ainsi l'intensité lumineuse qui lui parvient. Si $\mathbf{x}(t)$ est une paramétrisation de ce rayon où t représente la distance par rapport à l'œil de l'observateur, l'intensité lumineuse I qui parvient

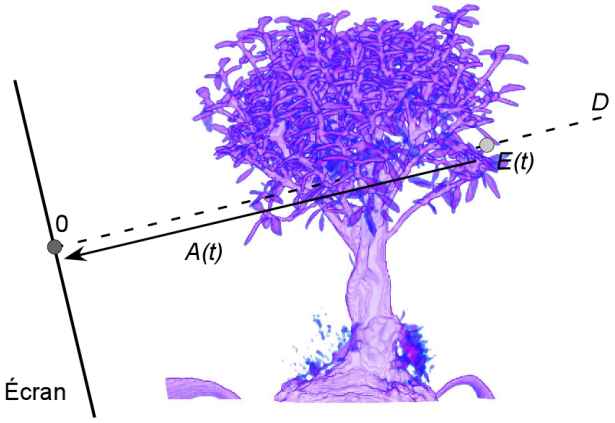

FIG. 2.6 – Principe du modèle d'*émission-absorption* en rendu volumique. L'émission d'intensité lumineuse $E(t)$ au point d'abscisse t (gris clair) sur le rayon est partiellement absorbée ($A(t)$) jusqu'au pixel de l'écran (gris foncé).

à l'observateur est définie par l'*intégrale de rendu volumique* :

$$\left| \begin{array}{l} I \;=\; \displaystyle\int_0^D E(t) \cdot e^{-A(t)}\, dt \\ \\ \text{avec :} \left\{ \begin{array}{rcl} E(t) & = & e\big(\mathbf{x}(t)\big) \\ \\ A(t) & = & \displaystyle\int_0^t a\big(\mathbf{x}(t')\big)\, dt' \end{array} \right. \end{array} \right. \tag{2.4}$$

où D est la distance au point où le rayon quitte le volume Ω, $e(\mathbf{x})$ et $a(\mathbf{x})$ respectivement l'intensité lumineuse émise et absorbée au point $\mathbf{x}$. Comme l'illustre la Figure 2.6, le terme $E(t)$ simule l'émission ponctuelle de rayonnement en chaque particule et le terme $A(t)$ son atténuation par absorption liée aux particules rencontrées entre le point d'émission et l'œil de l'observateur (ou le pixel de l'écran).

Bien que largement répandue, la méthode de visualisation fondée sur l'intégrale le long du rayon n'est pas exclusive en rendu volumique. Il existe en effet des alternatives, comme la méthode de *projection de l'intensité maximale* ou *MIP* (de l'anglais "Maximum Intensity Projection") qui conserve le long du rayon uniquement l'intensité d'émission maximale :

$$I \;=\; \max_{[0,D]}\big(E(t)\big) \tag{2.5}$$

Dans ce cas, il n'est ni question d'absorption ni même d'intégration, ce qui rend l'approche attractive par sa simplicité. Elle est principalement utilisée en imagerie médicale pour visualiser les données d'angiographie. La Figure 2.7 compare les images produites par intégration le long du rayon et par projection de l'intensité maximale sur des données de tomographie de vaisseaux sanguins à l'intérieur d'un crâne humain. De même, l'autre approche répandue est la méthode dite *Rayons X* qui effectue une somme des intensités le long du rayon sans atténuation, ce qui revient à appliquer un modèle d'*émission seule* :

$$I \;=\; \int_0^D E(t)\, dt \tag{2.6}$$

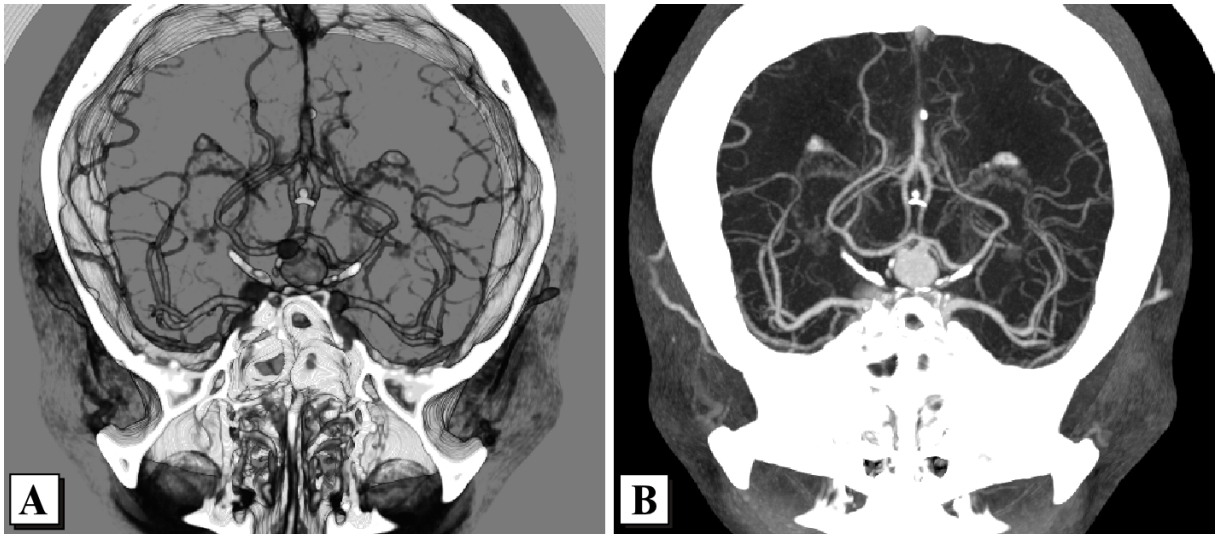

FIG. 2.7 – Comparaison du rendu volumique par intégration le long du rayon (A) et par *projection de l'intensité maximale* (B) sur des données de tomographie de vaisseaux sanguins à l'intérieur d'un crâne humain (d'après [REZK-SALAMA C. *et al.*, 2004]).

Ceci produit des images proches d'une radiographie aux rayons X. Par la suite, nous nous intéresserons exclusivement à la méthode classique d'intégration et toute référence au rendu volumique concernera cette méthode.

La forme brute de l'intégrale de rendu volumique de l'Équation (2.4) n'est pas adaptée à la visualisation d'un champ scalaire S continu dans l'espace. Dans ce cas, les propriétés d'émission et d'absorption en chaque point $\mathbf{x}$ du volume Ω dépendent de la valeur $S(\mathbf{x})$ du champ en ce point, en pratique de son approximation $s(\mathbf{x})$. Une étape de *classification* applique une *fonction de transfert* qui, pour chaque valeur s du champ scalaire, retourne les intensités d'*émission propre* et d'absorption correspondantes [PFISTER H. *et al.*, 2001]. À l'étape de classification, il est important d'insister sur la notion d'émission propre. En effet, l'émission au niveau d'une particule ne dépend pas seulement du rayonnement propre qu'elle produit, mais également de son environnement. Tandis que le rayonnement propre est défini par la seule étape de classification, une étape dite d'*illumination* (*"lighting & shading"* en anglais) tient compte des modifications liées à l'environnement de la particule, comme les interactions avec des sources lumineuses artificielles. Nous considérons pour le moment l'absence d'illumination, c'est-à-dire que l'émission propre issue de la classification ne subit pas de modifications environnementales. Nous verrons au Chapitre 3 comment des modèles d'illumination locale peuvent améliorer la qualité du rendu et ainsi la compréhension du champ scalaire.

L'émission propre définit en réalité la couleur de la particule. Dans la littérature, elle est généralement appelée *couleur primaire*[7]. Idéalement, la fonction de transfert de l'émission propre doit retourner, pour chaque valeur scalaire s, une fonction d'intensité lumineuse continue dans le domaine des longueurs d'onde visibles. En pratique, elle est définie dans l'espace de couleurs RGB (de l'anglais $\{Red, Green, Blue\}$, pour les couleurs fondamentales rouge, vert et bleu) [FOLEY J.D. *et al.*, 1996]. La fonction de transfert de l'émission propre est donc une fonction de $\mathbb{R}$ dans $\mathbb{R}^3$ et l'intégration est calculée indépendamment sur chaque intensité de l'espace

[7] terme emprunté à la terminologie OpenGL [WOO M. *et al.*, 1999] pour faire référence à la couleur avant illumination [KRAUS M. et ERTL T., 2005]

des couleurs. Cette fonction sera notée $c(s)$ et, par la suite, nous ferons référence à l'intensité lumineuse d'une manière générale. De même, l'absorption dépend théoriquement de la longueur d'onde considérée. De manière simplifiée, la fonction de transfert de l'absorption est une fonction de $\mathbb{R}$ dans $\mathbb{R}$ que nous noterons $\tau(s)$ et qui définit une absorption constante, indépendante de la longueur d'onde. En pratique, du point de vue de l'utilisateur, une seule fonction de transfert est définie de $\mathbb{R}$ dans $\mathbb{R}^4$ qui retourne, pour chaque valeur s du champ scalaire un 4-uplet dans l'espace $RGBA$ où RGB est l'émission propre de la particule et A son absorption. Les termes $E(t)$ et $A(t)$ de l'intégrale de l'équation (2.4) prennent donc la forme suivante, plus adaptée au cas de visualisation d'un champ scalaire :

$$\left| \begin{array}{rcl} E(t) & = & c\big(s(t)\big) \\ \\ A(t) & = & \displaystyle\int_0^t \tau\big(s(t')\big)\ dt' \end{array} \right. \tag{2.7}$$

L'édition de la fonction de transfert par l'utilisateur est une tâche non triviale, aussi des techniques de génération semi-automatique ont été mises au point pour pallier à cette difficulté [Kindlmann G. et Durkin J., 1998]. D'autre part, il est intéressant dans certains cas de faire dépendre l'émission propre et l'absorption de la particule non seulement de la valeur du champ scalaire mais également de la norme du gradient, ce qui permet de mettre en évidence les interfaces entre matériaux significativement différents [Levoy M., 1988]. La fonction de transfert est alors une fonction bidimensionnelle de $\mathbb{R}^2$ dans $\mathbb{R}^4$. L'interactivité s'en trouve d'autant plus complexe à mettre en œuvre et des outils d'édition avancée doivent être proposés pour faciliter le travail de l'utilisateur [Kniss J. *et al.*, 2002a].

Intégration numérique

L'expression obtenue ne peut être résolue analytiquement. L'intégration numérique suppose une discrétisation du rayon en n segments d'égale longueur $\Delta = D/n$ depuis l'œil de l'observateur jusqu'au point d'abscisse D sur le rayon. L'approximation par une somme de Riemann des intégrales dans les Équations (2.4) et (2.7) conduit à l'expression de I suivante :

$$\left| \begin{array}{l} I \ \approx \ \displaystyle\sum_{i=0}^{n-1} E(i\Delta)\ \cdot\ e^{-A(i\Delta)}\ \Delta \\ \\ \text{avec :} \left\{ \begin{array}{rcl} E(i\Delta) & = & c\big(s(i\Delta)\big) \\ \\ A(i\Delta) & \approx & \displaystyle\sum_{j=0}^{i-1} \tau\big(s(j\Delta)\big)\ \Delta \end{array} \right. \end{array} \right. \tag{2.8}$$

Cette expression se simplifie comme suit :

$$I \ \approx \ \sum_{i=0}^{n-1} c\big(s(i\Delta)\big) \prod_{j=0}^{i-1} e^{-\tau\left(s(j\Delta)\right)\Delta}\ \Delta \tag{2.9}$$

Si Δ est "suffisamment petit", les facteurs exponentiels peuvent être approximés par les deux premiers termes de leur développement de Taylor ($e^{-x} \approx 1 - x$) comme suit :

$$e^{-\tau\left(s(j\Delta)\right)\Delta} \ \approx \ 1 - \tau\big(s(j\Delta)\big)\Delta \tag{2.10}$$

ce qui conduit à :

$$I \approx \sum_{i=0}^{n-1} c\big(s(i\Delta)\big) \prod_{j=0}^{i-1} \Big(1 - \tau\big(s(j\Delta)\big)\Delta \Big) \Delta \tag{2.11}$$

Introduisons maintenant C_i et α_i, respectivement l'émission et l'absorption, également appelée *opacité*, du segment i. Ils peuvent être approximés comme suit :

$$\left| \begin{array}{rcl} C_i & \approx & c\big(s(i\Delta)\big)\ \Delta \\ \\ \alpha_i & \approx & \tau\big(s(i\Delta)\big)\ \Delta \end{array} \right. \tag{2.12}$$

L'association des Équations (2.11) et (2.12) aboutit à la version finale de l'intégrale de rendu volumique discrète :

$$I \approx \sum_{i=0}^{n-1} C_i \prod_{j=0}^{i-1} (1 - \alpha_j) \tag{2.13}$$

L'intégrale de rendu volumique discrète est donc une somme dont chaque terme correspond à la contribution d'un segment (émission ponctuelle et atténuation le long du rayon). La contribution de chaque segment peut être accumulée récursivement pour obtenir la valeur finale de l'intégrale. Ces étapes, dites de *composition*, peuvent s'effectuer d'arrière en avant ou inversement. Selon le sens, la *formule de composition*, également appelée *opérateur "over"* [Porter T. et Duff T., 1984], est différente. Pour une composition de l'arrière vers avant, l'opérateur "over" est défini comme suit :

$$C'_i = C_i + (1 - \alpha_i)\ C'_{i+1} \tag{2.14}$$

où C'_i est l'émission accumulée entre les segments $n-1$ et i. À noter que dans ce cas l'opacité n'a pas besoin d'être accumulée, chaque étape de composition faisant uniquement intervenir l'opacité du segment courant. Dans le cas d'une composition de l'avant vers l'arrière, l'opérateur "over", quelquefois appelé *"under"* pour marquer la différence [Ikits M. *et al.*, 2004], devient :

$$\left| \begin{array}{rcl} C''_i & = & C''_{i-1} + (1 - \alpha''_{i-1})\ C_i \\ \\ \alpha''_i & = & \alpha''_{i-1} + (1 - \alpha''_{i-1})\ \alpha_i \end{array} \right. \tag{2.15}$$

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

Les étapes de composition sont associatives mais non commutatives. Conséquence de l'associativité, il est possible de composer des groupes de segments séparément puis de composer les résultats. En revanche, la non commutativité implique que l'ordre de composition ne peut être modifié. Nous verrons par la suite que ces deux points auront des implications importantes sur les techniques de rendu volumique.

Le point commun de toutes les techniques de rendu volumique est le calcul de l'intégrale de l'Équation (2.4) pour chaque pixel de l'écran en considérant un rayon virtuel qui passe par le pixel et qui traverse le volume dans la direction d'observation. Ce calcul repose sur la discrétisation proposée à l'Équation (2.13) et procède récursivement à l'aide de l'un des opérateurs (2.14) ou (2.15) suivant le sens de parcours le long du rayon. Du point de vue du rayon virtuel, il est possible de distinguer les techniques de *discrétisation explicite* des techniques de *discrétisation implicite*. Cette distinction porte sur la façon de décomposer le travail en sous-tâches. Dans les techniques explicites, c'est chaque pixel de l'image finale qui est considéré indépendamment et pour lequel

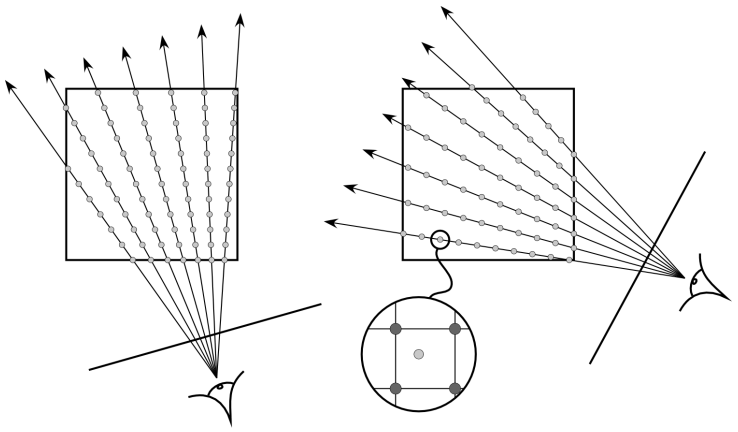

FIG. 2.8 – Discrétisation explicite de l'intégrale de rendu volumique par lancer de rayons. L'agrandissement montre que les points d'échantillonnage du rayon (gris clair) ne coïncident pas forcément avec les voxels de la grille (gris foncé).

est calculée la contribution du volume dans son ensemble par discrétisation explicite d'un rayon. Au contraire, dans les techniques implicites, le volume est décomposé et la contribution de chaque élément à l'image finale est calculée indépendamment, les rayons étant ainsi discrétisés collectivement et implicitement. Dans la littérature anglophone, cette distinction apparaît sous les termes d'*"image-order"* ou *"backward-mapping"* pour les techniques explicites, et d'*"object-order"* ou *"forward-mapping"* pour les techniques implicites [KAUFMAN A. et MUELLER K., 2005].

Discrétisation explicite par lancer de rayons depuis l'espace de l'image

La technique de décomposition dans l'espace de l'image la plus courante est le *lancer de rayons* [LEVOY M., 1988]. Dans ce cas, la discrétisation de l'intégrale de rendu est explicite dans la mesure où, comme le montre la Figure 2.8, un rayon virtuel est lancé explicitement dans le volume pour chaque pixel de l'espace de l'image et l'intégration est calculée le long de ce rayon par échantillonnage à intervalles réguliers. Comme le montre l'agrandissement sur cette même figure, les points d'échantillonnage du rayon ne coïncident pas forcément avec ceux de la grille. Ainsi, une étape d'*interpolation* applique un des filtres de reconstruction présentés en Section 2.1. La valeur interpolée subit ensuite l'étape de classification qui applique la fonction de transfert, puis la composition le long du rayon peut avoir lieu.

Il existe différents types de classification selon qu'elle intervient avant ou après interpolation (Figure 2.9). La *post-classification* applique la fonction de transfert sur une valeur interpolée du champ scalaire. Inversement, la *pré-classification* applique la fonction de transfert aux voxels de la grille, l'interpolation ayant lieu dans l'espace *RGBA*. Cette dernière méthode conduit dans certains cas à des incorrections sur l'image finale. Une des raisons est que chaque composant de l'espace *RGBA* est interpolé indépendamment [WITTENBRINK C.M. *et al.*, 1998]. Ces effets sont évités en multipliant l'émission propre par l'absorption préalablement à l'interpolation [WITTENBRINK C.M. *et al.*, 1998, GASPARAKIS C., 1999]. Les couleurs primaires ainsi utilisées sont dites *couleurs associées* (*"associated colors"* ou *"opacity-weighted colors"* en anglais) [BLINN J.F., 1994]. Cependant, une autre source d'incorrections réside dans la non linéarité de la fonction de transfert [ENGEL K. *et al.*, 2001]. La Figure 2.9 montre que seule une fonction de transfert constante ou linéaire conduirait à un résultat correct, ce qui dans la pratique est peu probable. Dans ce cas, seule une post-classification produit le résultat correct au sens où il

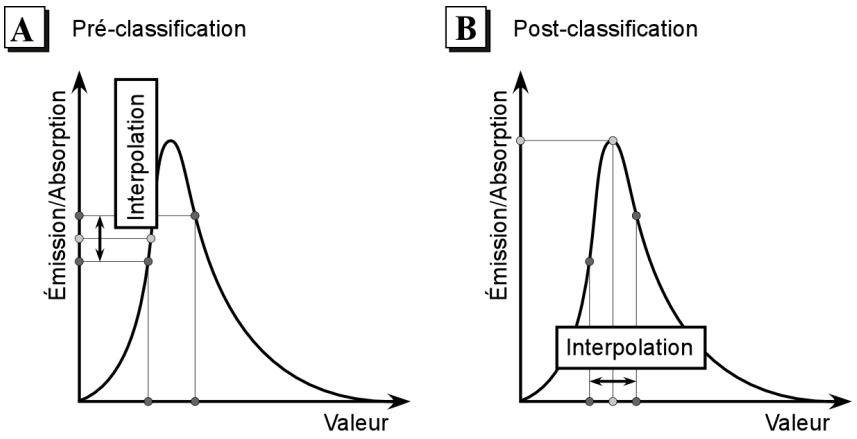

FIG. 2.9 – Différence entre pré-classification (A) et post-classification (B). Selon que l'étape de classification intervient avant (pré-classification) ou après (post-classification) l'étape d'interpolation, les propriétés d'émission/absorption retournées par la fonction de transfert peuvent être très différentes (d'après [REZK-SALAMA C., 2001]).

s'agit d'appliquer une fonction de transfert à un champ scalaire continu défini sur un maillage [ENGEL K. *et al.*, 2001, LICHTENBELT B. *et al.*, 1998]. Certaines applications telles que la médecine ont cependant recours aux techniques de pré-classification pour séparer des organes lors d'étapes de prétraitement du volume.

Indépendamment de l'ordre d'application des opérations d'interpolation et de classification, le lancer de rayons permet l'intégration directe de méthodes d'optimisations [LEVOY M., 1990] telles que la *terminaison précoce de rayon* (*"early ray termination"* en anglais) et l'*accélération dans les espaces vides* (*"empty space skeeping"* en anglais). La première consiste à parcourir le rayon de l'avant vers l'arrière et à stopper la progression dès qu'un degré "suffisamment élevé" d'opacité a été atteint, la limite pouvant être fixée par l'utilisateur. La seconde utilise des structures de données hiérarchiques qui permettent d'accélérer la progression le long du rayon dans les zones transparentes du volume.

Discrétisation implicite par projection depuis l'espace de l'objet

"Splatting" Les techniques de décomposition dans l'espace de l'objet sont appelées techniques de *projection*. Elles ont pour philosophie l'approche originale du *"splatting"* [WESTOVER L., 1989]. Cette approche, illustrée à la Figure 2.10, calcule la contribution de chaque voxel de la grille indépendamment par projection sur le plan de l'image. Ainsi, les voxels sont successivement composés au niveau des pixels qu'ils affectent. L'extension sur l'image de l'*empreinte* (*"footprint"* en anglais) du voxel projeté détermine les pixels affectés. Elle dépend d'un noyau de convolution 3D appliqué au voxel avant projection. Il s'agit généralement d'un noyau Gaussien, équivalent à une petite sphère diffuse autour du voxel. La symétrie du noyau permet une préintégration de son empreinte 2D une seule fois pour tous les voxels indépendamment du point de vue. Il suffit ensuite de multiplier cette empreinte générique par la valeur du voxel projeté, ce qui fait l'efficacité de la méthode.

Du fait de la non commutativité de la composition, l'ordre de projection est important. Les voxels sont classés par tranches alignées selon l'axe du volume le plus perpendiculaire au plan de l'image et, dans chaque tranche, par ordre croissant de leur distance à l'œil de l'observateur.

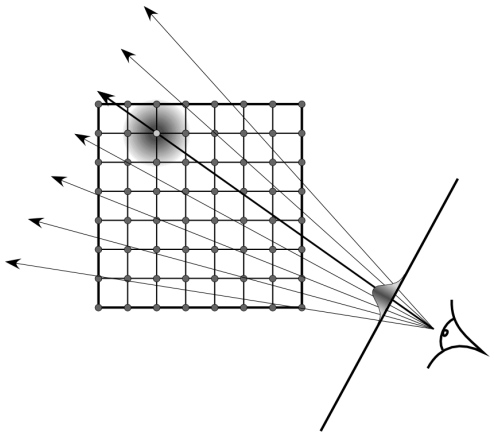

FIG. 2.10 – Discrétisation implicite de l'intégrale de rendu volumique par la technique du *"splatting"*. La contribution de chaque voxel à l'image finale est calculée par projection. L'empreinte (*"footprint"* en anglais) du voxel projeté (gris clair) sur l'image dépend d'un noyau de convolution (gaussien dans notre cas) appliqué avant projection.

Cette approche appelée *"axis-aligned sheet buffered splatting"* [WESTOVER L., 1990], permet de projeter chaque tranche indépendamment sur le plan de l'image, puis de les composer. Pour aller plus loin, les voxels peuvent être classés par tranches, non pas alignées sur un axe du volume, mais parallèlement à l'axe de l'image. Cette méthode appelée *"image-aligned sheet buffered splatting"* [MUELLER K. et CRAWFIS R., 1998] élimine certains artefacts liés à l'approche précédente. Nous verrons comment les techniques de rendu volumique à base de textures (Section 2.3.2) s'apparentent à ces méthodes.

La distinction entre pré-classification et post-classification soulignée dans la section précédente pour les méthodes explicites intervient également pour le "splatting". L'approche originale [WESTOVER L., 1989, WESTOVER L., 1990] applique la fonction de transfert avant projection, ce qui s'apparente à une pré-classification. Une autre approche, plus récente, permet d'effectuer une post-classification dans l'espace de l'image après projection [MUELLER K. *et al.*, 1999].

"Shear-Warp" Similaire au "splatting" dans sa façon de procéder par projection depuis l'espace de l'objet, la technique du *"shear-warp"* [LACROUTE P. et LEVOY M., 1994] est souvent considérée comme une méthode hybride relevant aussi bien du lancer de rayons que de la projection. Nous la classons cependant parmi les méthodes implicites de projection car les méthodes de projection à base de textures que nous verrons par la suite (Section 2.3.2) se sont très largement inspirées de cette approche.

Le "shear-warp" part du principe que l'élément coûteux dans le lancer de rayons est le grand nombre d'interpolations trilinéaires impliquées lors de la discrétisation de chaque rayon. Ainsi, l'idée originale est de remplacer ces interpolations trilinéaires dans le volume de la grille par des interpolations bilinéaires dans ses plans. Pour cela, les points de discrétisation le long du rayon sont placés intelligemment sur les plans de la grille les plus perpendiculaires au plan de l'image. D'autre part, comme le montre la Figure 2.11, la composition le long du rayon est calculée sur une image intermédiaire dans une grille transformée par *cisaillement*/*cisaillement-homothétie* (*"shear"*/*"shear-scale"* suivant le mode de vision parallèle ou perspective) le long de ces mêmes plans. L'alignement des voxels qui résulte de cette transformation permet d'effectuer la composition par simple projection des plans de la grille sur l'image intermédiaire, avec interpo-

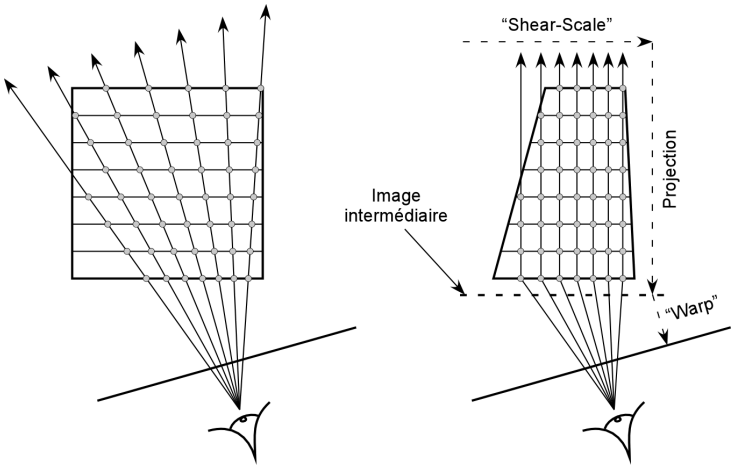

FIG. 2.11 – Discrétisation implicite de l'intégrale de rendu volumique par la technique du *"shear-warp"*. Après *cisaillement-homothétie* (*"shear-scale"*) du volume, la composition est calculée par projection sur une image intermédiaire distordue puis *plaquage* (*"warp"*) sur l'image finale.

lation bilinéaire dans les plans. L'image intermédiaire distordue ainsi obtenue est ensuite *plaquée* (*"warp"*) sur l'image finale par une transformation 2D. Ainsi, en combinant un cisaillement 1D suivi d'un plaquage 2D et quelques structures d'encodage du volume et de l'image intermédiaire, le "shear-warp" est la technique logicielle de rendu volumique la plus efficace.

Une dernière remarque sur le "shear-warp" est la nécessité de trier les voxels selon l'axe de cisaillement. Cet axe peut être amené à varier lorsque le point de vue est modifié. Ainsi, pour une manipulation optimale il est nécessaire de stocker trois copies du volume de données chacune organisée selon un axe particulier. Les structures d'encodage proposées dans l'implantation originale [LACROUTE P. et LEVOY M., 1994] permettent cependant de réduire le coût mémoire occasionné par cette copie.

Pour conclure sur les techniques de visualisation volumique, il est important de préciser que les méthodes directes ont longtemps souffert de leur temps de calcul qui les pénalisait par rapport aux méthodes indirectes où l'essentiel du calcul intervient dans une étape de prétraitement destinée à extraire un sous-ensemble du volume Ω. Il est en effet nécessaire, pour une bonne compréhension de la structure dans l'espace du champ scalaire, de modifier le point de vue en temps réel, ce qui donne de la profondeur au rendu [SOLLENBERG R.L. et MILGRAM P., 1993]. Or, chaque image générée à partir d'un point de vue différent doit être calculée intégralement. Cependant, le développement de techniques logicielles toujours plus performantes à l'image du "shear-warp" a permis aux méthodes directes de garder une véritable crédibilité aux yeux de certains. D'autre part, le développement du matériel graphique des dix dernières années leur a donné un second souffle et les techniques de rendu volumique modernes supportant l'accélération matérielle génèrent des images en temps réel. Nous allons donc maintenant nous intéresser à ces méthodes plus récentes.

2.3 Visualisation volumique et accélération matérielle

Nous aborderons dans cette section l'accélération matérielle des méthodes directes de visualisation volumique. Cependant, avant de voir les techniques plus en détail, nous allons commencer par présenter le matériel graphique standard et l'évolution fulgurante qu'il a connu depuis le début des années 2000. En dernier lieu, nous parlerons brièvement du développement de matériel graphique spécialisé, dédié au rendu volumique de grilles régulières.

2.3.1 Matériel graphique standard

En plus d'être en charge de l'exécution des applications telles que les modeleurs 3D, le *processeur central* de l'ordinateur, ou *CPU* (de l'anglais "Central Processing Unit"), a longtemps pris en charge l'ensemble des opérations d'affichage. C'est le cas par exemple des algorithmes de rendu volumique présentés à la Section 2.2.2. L'apparition des stations graphiques au milieu des années 1990 a permis de déléguer en partie cette charge à une puce spécialisée. Son évolution au cours des dix dernières années a conduit à un véritable *processeur graphique*, ou *GPU* (de l'anglais "Graphics Processing Unit", disposant de son propre espace mémoire sur la *carte graphique*. La Figure 2.12 présente en détails le principe de fonctionnement et l'évolution de ce dernier depuis le début des années 2000. Les étapes successives de traitement intervenant dans le GPU sont communément appelées le *pipeline graphique* [Foley J.D. *et al.*, 1996, Möller T. et Haines E., 2002]. Notons que son architecture a été clairement pensée pour le développement d'applications telles que les jeux vidéo dans lesquelles il s'agit essentiellement de visualiser des objets 3D représentés sous la forme de surfaces complexes. Son rôle est de prendre en entrée une représentation tridimensionnelle de ces objets et d'afficher en sortie une image bidimensionnelle à l'écran en fonction de l'état de l'environnement dans lequel ils sont plongés. L'application communique avec le GPU à travers une interface de programmation ou *API* (de l'anglais "Application Programming Interface") spécialisée telle que le standard multi-plateforme OpenGL [Segal M. et Akeley K., 2003] ou l'API propriétaire de Microsoft Direct3D [Microsoft Corporation, 2002a]. Pour des raisons de portabilité du code, les travaux présentés dans ce mémoire ont été développés exclusivement avec OpenGL, standard de développement d'applications scientifiques.

Comme le montre la Figure 2.12, le pipeline graphique prend en entrée un flux de *sommets* dans l'espace, lesquels peuvent définir des polygones, des lignes ou de simples points isolés. Une surface tridimensionnelle complexe doit donc être décomposée en un ensemble de polygones par le CPU dont les sommets sont envoyés par groupes sur le GPU. Ce dernier assemble les groupes de sommets sous la forme de *primitives*. Cette représentation est encore très proche de celle que manipule le CPU à la différence près que les polygones de plus de trois sommets ont été triangulés pour former plusieurs primitives. Cependant, l'image bidimensionnelle finale stockée dans la *mémoire écran* (*"framebuffer"* en anglais) est représentée sous la forme d'une matrice de *pixels*. Entre les primitives et les pixels, le GPU manipule des éléments intermédiaires appelés *fragments* qui peuvent être considérés comme l'équivalent 3D des pixels. En effet, pour chaque pixel affecté par une primitive, l'étape de *rasterisation* génère un fragment dans l'espace. Ainsi, si plusieurs primitives affectent un même pixel, plusieurs fragments sont générés indépendamment, et la couleur finale du pixel tient compte de chacun d'eux et de leur position respective dans l'espace grâce aux *opérations de fragment*. Par exemple, l'utilisation du *z-buffer* permet de stocker la profondeur du dernier fragment et de gérer les occlusions en ne tenant pas compte d'un fragment plus profond. D'autre part, la couleur *RGB* de chaque fragment peut être mélangée

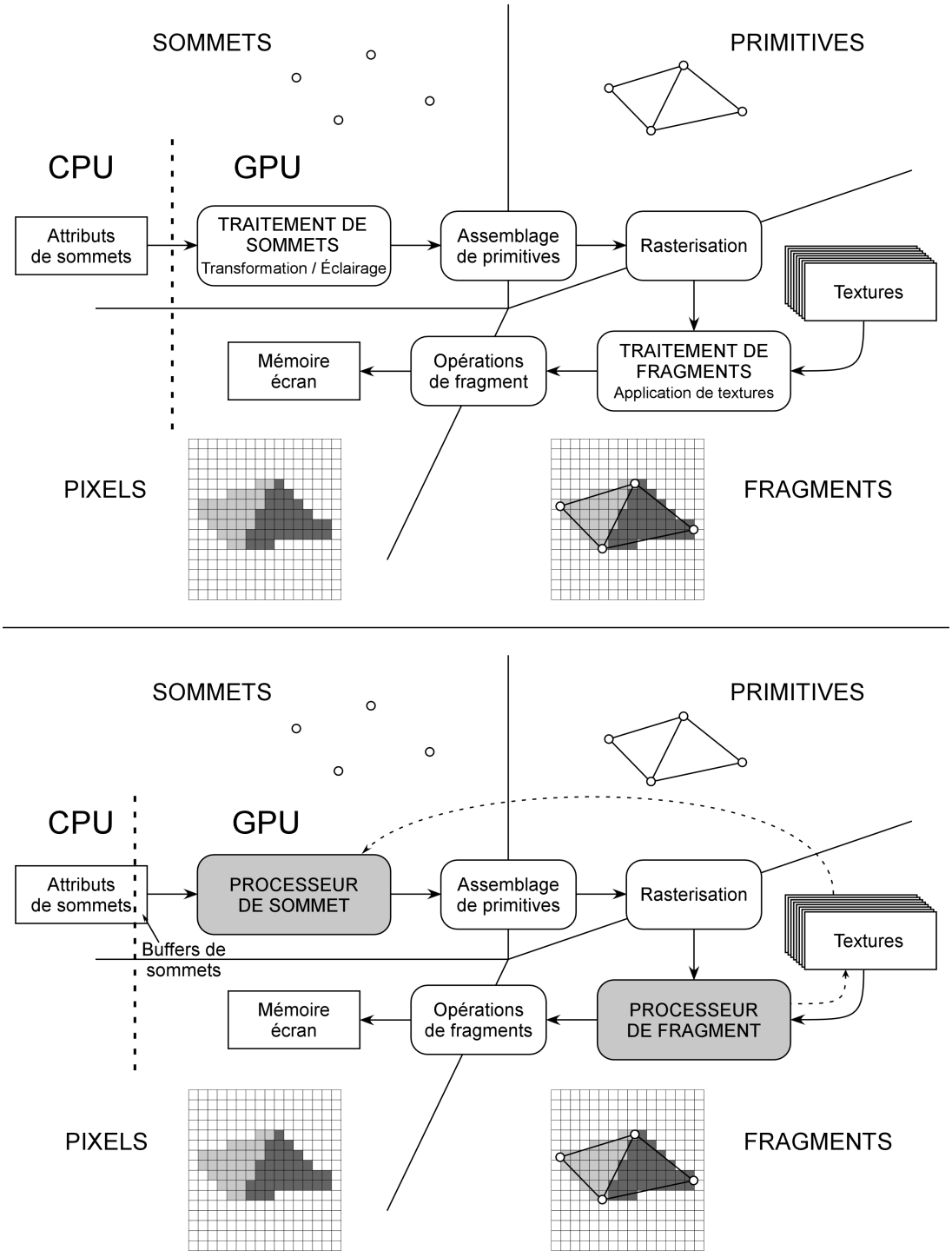

FIG. 2.12 – Principe de fonctionnement simplifié du *pipeline graphique* classique implanté dans le matériel graphique standard (haut) et son évolution vers une plus grande flexibilité depuis le début des années 2000 (bas).

avec la couleur courante du pixel en fonction d'un paramètre *alpha*. Cette technique s'appelle l'*"alpha-blending"* et repose sur l'utilisation de couleurs $RGBA$ ou $RGB\alpha$.

Il est important de noter l'existence de deux unités de traitement essentielles dans le pipeline graphique classique : l'*unité de traitement de sommets* et l'*unité de traitement de fragments* (Figure 2.12). L'unité de traitement de sommets est en charge des opérations de transformation entre l'espace 3D de l'application dans lequel est représenté le modèle et l'espace 2D de l'écran. D'autre part, c'est également à ce niveau qu'ont lieu les calculs d'illumination, simulant ainsi l'interaction du modèle avec des sources lumineuses virtuelles placées dans son environnement. Toutes ces opérations sont effectuées sur chaque sommet de manière indépendante et le résultat est interpolé linéairement à l'étape de rasterisation. Chaque fragment ainsi généré est traité individuellement par l'unité de traitement de fragments. Celle-ci consiste en un ensemble d'unités de travail, appelées *"pixel pipelines"*, qui traitent les fragments en parallèle. À cette étape, une couleur de *texture* est éventuellement appliquée au fragment. Une texture est une image 2D stockée sous forme de pixels (*texels*[8] dans ce contexte) en mémoire graphique et plaquée sur les primitives en fonction de coordonnées 2D définies sur les sommets et interpolées linéairement au niveau des fragments. La couleur à appliquer au fragment est obtenue par interpolation bilinéaire dans la texture basée sur ses coordonnées de texture. Cette technique appelée *plaquage de texture* (*"texture mapping"* en anglais) permet de représenter de fines variations de couleurs sans pour autant densifier la géométrie, ce qui dégraderait les performances. Plusieurs textures peuvent être utilisées à la fois, la carte graphique possédant un nombre limité d'*unités de texture*. Bien qu'ayant permis une évolution significative des performances à la fin des années 1990, ce pipeline figé dans le silicium laisse peu de liberté au programmeur d'applications graphiques qui doit recourir au CPU dès lors qu'il désire implanter des effets originaux.

Avec le développement du matériel graphique standard, les unités de traitement de sommets et de fragments ont donc progressivement évolué pour aboutir aujourd'hui à de véritables processeurs programmables : respectivement le *processeur de sommet* et le *processeur de fragment* sur la Figure 2.12. Ces processeurs exécutent des programmes spécifiés par l'application graphique et écrits dans un langage spécialisé, les *programmes de sommet* et *programmes de fragment*. Aux premiers langages assembleurs définis dans les extensions OpenGL `ARB_vertex_program` et `ARB_fragment_program` [Kilgard M.J., 2003] ont succédé des langages haut niveau tels que l'OpenGL Shading Language [Kessenich J. *et al.*, 2003, Rost R., 2004], le langage propriétaire de NVIDIA Cg [Mark W.R. *et al.*, 2003, Fernando R. et Kilgard M.J., 2003] ou celui de Microsoft HLSL [Microsoft Corporation, 2002b]. En raison de leur architecture et des contraintes de performances associées, les processeurs graphiques représentent donc aujourd'hui une formidable puissance de calcul parallèle aussi bien pour programmer des effets purement graphiques que pour le calcul haute performance [Fernando R., 2004, Pharr M., 2005].

Les éléments déterminants qui ont ouvert la voie de l'accélération matérielle au rendu volumique ont été dans un premier temps l'introduction d'architectures supportant le plaquage de texture 2D puis 3D telles que les stations graphiques Reality Engine de SGI, et plus récemment la flexibilité des processeurs graphiques programmables disponibles sur le matériel graphique "grand public". Nous allons donc voir comment les techniques de rendu volumique ont tiré parti de ces différentes avancées matérielles.

[8] contraction de l'anglais "texture elements" (éléments de texture)

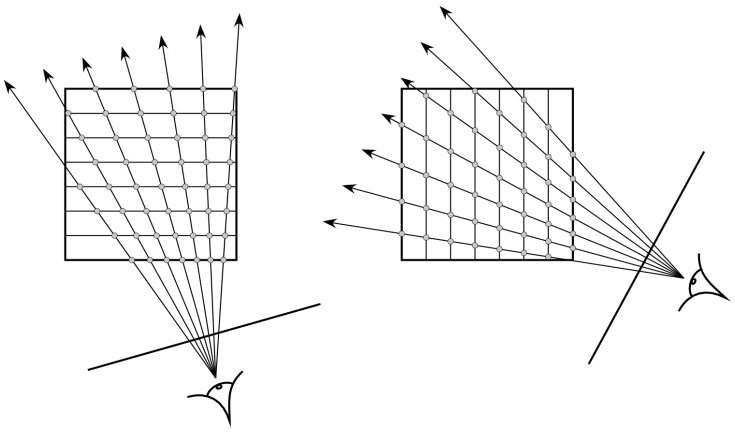

FIG. 2.13 – Accélération matérielle du rendu volumique à base de textures 2D. Le volume est stocké en mémoire graphique sous forme de trois séries de textures 2D perpendiculaires à chaque axe. À chaque instant, la série de textures perpendiculaires à l'axe le plus proche de la direction d'observation est plaquée sur une série de polygones puis composée par pixel grâce aux opérations d'"alpha-blending".

2.3.2 Visualisation volumique à base de textures

Comme nous l'avons précisé à la section précédente, l'un des éléments déterminants dans l'adoption du matériel graphique standard par les techniques de rendu volumique est l'introduction du support du plaquage de texture. En effet, l'utilisation de textures permet d'accélérer considérablement les opérations d'interpolation bilinéaire (textures 2D) et trilinéaire (textures 3D) [CABRAL B. *et al.*, 1994] particulièrement coûteuses en rendu volumique (Section 2.2.2). La façon dont est conçu le pipeline graphique (Section 2.3.1) nous conduit à aborder dans un premier temps l'accélération matérielle des approches implicites (approches objet), plus intuitive que pour les approches explicites (approches image).

Approches implicites

Textures 2D Les approches implicites à base de textures 2D [CABRAL B. *et al.*, 1994] procèdent de manière analogue à l'algorithme du "shear-warp". Les interpolations trilinéaires dans le volume sont remplacées par des interpolations bilinéaires dans ses plans. Ces plans sont stockés sous forme de textures 2D en mémoire graphique. Comme le montre la Figure 2.13, ces textures sont plaquées sur une série de polygones perpendiculaires à l'axe du volume le plus proche de la direction d'observation. Le GPU se charge des transformations géométriques dans l'unité de traitements de sommets, de la décomposition en fragments à l'étape de rasterisation et l'interpolation bilinéaire native du matériel graphique est utilisée dans l'unité de traitement de fragments. Cette dernière étant optimisée pour effectuer un très grand nombre d'interpolations par seconde, il est possible de générer les plans de fragments texturés en temps réel. Finalement, la composition est calculée par pixel grâce aux opérations d'"alpha-blending".

Cependant, comme le montre la Figure 2.13, lors d'un changement de point de vue l'axe du volume le plus proche de la direction de visualisation peut changer. Dans ce cas, la série de plans à dessiner et donc à stocker sous forme de textures est modifiée. Pour réagir interactivement à une telle éventualité, il est nécessaire de stocker le volume trois fois, sous la forme de trois séries de plans alignés perpendiculairement à chaque axe. D'autre part, au moment du changement de plans, des artefacts apparaissent, proches de ceux observés dans l'"axis-aligned sheet buffe-

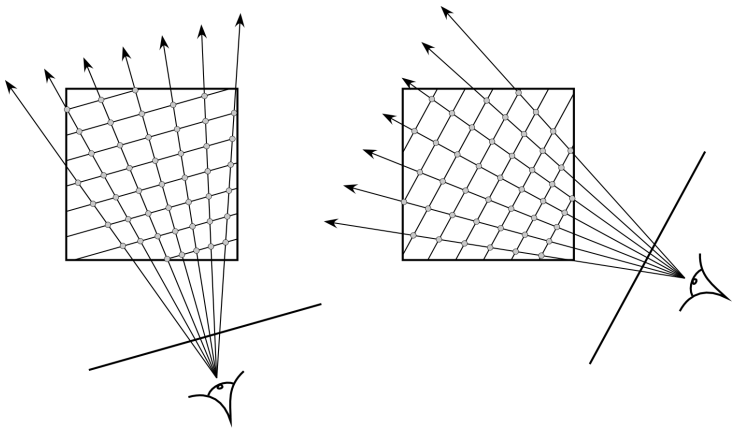

FIG. 2.14 – Accélération matérielle du rendu volumique à base de textures 3D. Le volume est stocké en mémoire graphique sous forme de texture 3D. L'interpolation trilinéaire native du matériel graphique permet de générer en temps réel des plans alignés sur l'image.

red splatting" [WESTOVER L., 1990] présenté à la Section 2.2.2. Ceux-ci peuvent être réduits en générant des plans intermédiaires à la volée sur le GPU [REZK-SALAMA C. *et al.*, 2000]. Malheureusement, cette approche ne résout pas le problème de manière satisfaisante et surtout ne permet pas de s'affranchir des trois copies du volume. Pour cela, il est nécessaire d'utiliser des textures 3D.

Textures 3D Les approches à base de textures 3D [CABRAL B. *et al.*, 1994] conservent l'idée de dessiner des plans texturés de l'arrière vers l'avant qui a conduit à l'utilisation de textures 2D. Cependant, l'interpolation bilinéaire dans les textures 2D, qui nécessitait l'utilisation de plans alignés selon les axes du volume, est remplacée par une interpolation trilinéaire dans les textures 3D. Ceci permet donc de dessiner des plans alignés sur l'image indépendamment de l'orientation du volume qui est stocké une seule fois sous forme de texture 3D. L'interpolation trilinéaire native du matériel graphique permet de générer de nouveaux plans en temps réel lorsque le point de vue est modifié. D'autre part, cette approche résout définitivement le problème des artefacts liés au changement de série de plans dans l'utilisation de textures 2D, de la même manière que l'"image-aligned sheet buffered splatting" [MUELLER K. et CRAWFIS R., 1998] présenté à la Section 2.2.2 apportait une solution aux artefacts de l'"axis-aligned sheet buffered splatting" [WESTOVER L., 1990].

L'une des principales limitations de l'utilisation des textures 3D par rapport aux textures 2D est le schéma d'accès aux textures qui réduit l'efficacité du cache optimisé pour les textures 2D [REZK-SALAMA C. *et al.*, 2004], beaucoup plus utilisées dans les applications "grand public" telles que les jeux vidéo. Cette distinction se justifie cependant de moins en moins avec les nouvelles cartes graphiques et nous verrons au Chapitre 6 qu'il est possible d'optimiser l'utilisation du cache en jouant sur la taille des textures 3D utilisées.

Approches explicites

Historiquement, l'approche explicite par lancer de rayons a toujours été considérée comme la référence en termes de qualité de rendu. En contrepartie, cette qualité s'est longtemps payée au prix de performances bien inférieures à celles des approches implicites à base de textures.

Pour cette raison, un effort particulier a été consenti pour faire bénéficier le lancer de rayons des avancées du matériel graphique. L'accélération matérielle dans ce cas est beaucoup plus complexe et moins intuitive à mettre en œuvre. Outre le recours aux textures 2D et 3D, ces techniques tirent également partie du caractère programmable des cartes graphiques, ce qui explique leur apparition plus tardive [Röttger S. *et al.*, 2003, Krüger J. et Westermann R., 2003].

La technique de lancer de rayons présente un parallélisme intrinsèque, chaque rayon pouvant être calculé indépendamment. Ainsi, le parallélisme de l'unité de traitement de fragments ("pixel pipeline") est directement exploité en associant un rayon à chaque pixel. Il suffit de dessiner un polygone qui va générer un fragment pour chaque pixel de l'écran d'où un rayon doit être lancé, un programme de fragment se chargeant de calculer l'intégrale de l'Équation (2.13). Comme nous l'avons vu précédemment (Section 2.2.2), ce calcul est itératif et se base sur les opérateurs (2.14) ou (2.15). La boucle est gérée au niveau du CPU par l'application qui génère autant de passes de rendu qu'il y a d'itérations. Comme pour les techniques implicites, les données volumiques sont stockées sous forme de textures 3D en mémoire graphique. D'autre part, l'information nécessaire à chaque rayon au cours de la propagation (couleur/opacité accumulée, ...) est stockée dans des textures 2D avec une correspondance directe entre texels et pixels de l'écran, ce qui permet au programme de fragment d'y accéder lors du rendu. À noter que le programme accède à ces textures en écriture, ce qui en pratique est mis en œuvre à l'aide de l'extension OpenGL `EXT_framebuffer_object` [Kilgard M.J., 2003] ou de la fonctionnalité *render-to-texture* de Direct3D [Microsoft Corporation, 2002a]. D'autre part, le matériel graphique ne supporte pas l'accès au textures simultanément en lecture et en écriture dans une même passe de rendu. Pour cette raison, une technique dite de *ping-pong* utilise deux copies de chaque texture 2D, l'une accédée en lecture, l'autre en écriture, leur rôle étant inversé à chaque nouvelle passe. Les optimisations classiques du lancer de rayons telles que la terminaison précoce de rayon et l'accélération dans les espaces vides (Section 2.2.2) peuvent également être mises au point dans ce contexte. La première repose sur l'utilisation du z-buffer tandis que l'équivalent des structures de données hiérarchiques de la seconde peut être stocké dans des textures 3D.

Dans nos travaux, nous avons privilégié une approche implicite à base de textures 3D qui reste plus performante que l'approche explicite. Le différentiel en termes de qualité de rendu observé sur les méthodes implicites tend à diminuer depuis l'apparition de techniques de rendu haute qualité [Rezk-Salama C. *et al.*, 2000, Engel K. *et al.*, 2001, Lum E.B. *et al.*, 2004] plus sophistiquées que l'approche classique. Nous aborderons ces aspects plus en détail au Chapitre 3 dans la mesure où notre approche repose largement sur ces travaux préliminaires.

2.3.3 Matériel graphique spécialisé

Comme nous l'avons vu en Section 2.3.1 et bien qu'elle puisse être détournée pour l'usage d'applications scientifiques, l'architecture du matériel graphique standard vise en premier lieu à optimiser les applications "grand public". D'autre part, les grandes évolutions qu'elle a connues pour aboutir au niveau de flexibilité actuel n'ont été initiées qu'au début des années 2000. Ainsi, les années 1990 ont été le théâtre de travaux visant à mettre au point des architectures graphiques spécialement destinées à l'accélération du rendu volumique. Un aperçu de ces architectures est proposé dans [Ray H. *et al.*, 1999], et plus récemment par l'un des pionniers en la matière dans [Kaufman A. et Mueller K., 2005]. Il existe trois principaux fronts : le projet VIRIM de l'Université de Mannheim, le projet VIZARD de l'Université de Tübingen, tous deux en Allemagne, et enfin le projet Cube de l'Université d'État de New York (SUNY) à Stony Brook

aux États-Unis. Chacun de ces projets vise à mettre au point des architectures qui implantent le lancer de rayons en natif sur le matériel graphique.

VIRIM [Günther T. *et al.*, 1994] est une architecture composée de deux unités : une unité de géométrie chargée des opérations d'interpolation et de calcul de gradient dans le volume, et une unité de lancer de rayons chargée de la composition. VIRIM est capable de visualiser un volume de $256^2 \times 128$ voxels à 2,5 fps[9]. L'agrégation de 16 modules permet d'atteindre 10 fps sur un volume de même taille.

VIZARD-II [Meissner M. *et al.*, 1998] vise entre autres à réduire l'utilisation de la bande passante mémoire par des techniques telles que la compression de données. Elle est composée de quatre unités : une unité de contrôle chargée des calculs d'intersection, une unité de mémoire chargée du stockage, une unité d'interpolation (trilinéaire) et une unité d'illumination/composition. Cette architecture a également recours à des optimisations du lancer de rayons comme la terminaison précoce de rayon (Section 2.2.2). Les performances estimées pour un volume de 256^3 sont de 10 fps en moyenne.

Cube, le plus abouti des trois projets, reposait dans sa version initiale Cube-1 sur une architecture mémoire optimisée [Kaufman A. et Bakalash R., 1988]. Les améliorations successives du système, Cube-2 [Bakalash R. *et al.*, 1992], Cube-3 [Pfister H. *et al.*, 1994] et Cube-4 [Pfister H. et Kaufman A., 1996], ont abouti à EM[10]-Cube [Osborne R. *et al.*, 1997], premier pas vers une version commerciale de Cube-4. La commercialisation s'est concrétisée en 1999 avec la carte VolumePro 500 [Pfister H. *et al.*, 1999] de Mitsubishi Electric, capable de visualiser un volume de 256^3 à 30 fps. Cette dernière est basée sur une implantation native de l'algorithme du "shear-warp" pour optimiser l'accès aux données avec interpolation trilinéaire, contrairement à l'interpolation bilinéaire dans le "shear-warp" classique. Cette approche a ensuite été remplacée par une technique dite *"shear-image"* [Wu Y. *et al.*, 2003] qui permet d'éliminer la projection intermédiaire du "shear-warp" tout en conservant l'accès mémoire efficace. Combinée aux optimisations classiques du lancer de rayons (terminaison précoce et accélération dans les espaces vides, Section 2.2.2), cette architecture a été commercialisée par TeraRecon Inc. sous le nom de VolumePro 1000.

Du fait de son coût prohibitif, ce type de matériel spécialisé est utilisé dans des contextes particuliers, tels que des centres de recherche spécialisés ou des hôpitaux. D'autre part, son architecture dédiée se justifie de moins en moins face à l'augmentation constante des performances et de la flexibilité du matériel graphique standard présenté à la Section 2.3.1. Cependant, l'un comme l'autre révèlent rapidement leurs limites lorsque la taille des données à visualiser devient importante. En effet, au-delà d'un certain seuil, les calculs et éventuellement la quantité de mémoire de texture nécessaires à la synthèse de l'image en temps réel dépassent la capacité du matériel graphique même le plus performant. Dans certains contextes d'application et notamment en sismique, une sous-partie du volume total suffit à mener une interprétation correcte, point sur lequel nous aurons l'occasion de revenir au Chapitre 5. Malheureusement ceci ne fait que détourner le problème, la taille minimale exploitable d'un sous-volume pouvant elle-même s'avérer supérieure à la capacité de traitement d'un seul processeur. Dans ce cas, il est possible de recourir à l'adage "diviser pour régner" en calculant l'image finale sur un groupe de processeurs travaillant en parallèle.

[9] de l'anglais "frames per second" (images par seconde)
[10] de l'anglais "Enhanced Memory" (mémoire améliorée)

2.4 Visualisation volumique parallèle

Lorsque la taille d'un problème informatique quelconque devient supérieure à la capacité de traitement temps réel d'un seul processeur, la décomposition en sous-problèmes indépendants de taille inférieure traités en parallèle par un ensemble de processeurs permet de retrouver des performances acceptables. Il existe différentes stratégies de décomposition selon le contexte. Tout traitement informatique peut être défini comme l'application d'un certain nombre d'opérations sur un ensemble de données initiales pour aboutir à un résultat. Il est donc possible de spécialiser chaque processeur pour un sous-ensemble des opérations à effectuer et de propager un flux de données à travers les processeurs. Cette décomposition en tâches porte le nom de *parallélisme de contrôle*. Une alternative, dite *parallélisme de données*, consiste à affecter à chaque processeur un sous-ensemble des données initiales et à leur faire calculer les résultats en parallèle. Par exemple, le matériel graphique standard introduit à la Section 2.3.1 dispose d'un certain nombre d'unités de traitement spécialisées pour des opérations particulières et implante donc de manière intrinsèque un parallélisme de contrôle que le terme pipeline graphique illustre parfaitement. D'autre part, les "pixel pipelines" de l'unité de traitement de fragments permettent de superposer localement un parallélisme de données. Tout ceci correspond cependant à des implantations matérielles de très bas niveau auxquelles le programmeur n'a pas accès directement. À un niveau supérieur, la visualisation parallèle en général et le rendu volumique parallèle en particulier reposent sur un parallélisme de données qui se décline sous différentes formes. Nous verrons donc tout d'abord quelles sont les formes de parallélisme de données communément rencontrées en visualisation. Quelle que soit la stratégie, elle fait intervenir une *architecture multiprocesseur* à laquelle est associé un *modèle de programmation*, deux aspects déterminants dans la conception d'un système de rendu volumique parallèle sur lesquels il est important de s'attarder. Finalement, nous nous intéresserons à un problème précis d'importance capitale pour les performances du système que nous introduirons au cours de la Partie III : le partitionnement d'objet sur cluster de PCs.

2.4.1 Partitionnement et visualisation parallèle

En Section 2.2.2, les techniques de visualisation volumique directe ont été classées en deux grandes familles selon qu'elles procèdent depuis l'espace de l'image ou celui de l'objet. Il est tout naturellement possible d'appliquer la même distinction aux techniques de visualisation volumique parallèle. Ainsi, le parallélisme peut s'appliquer indifféremment dans l'espace de l'image ou celui de l'objet, ce qui conduit à l'existence de deux types de décomposition du problème : le *partitionnement d'image* et le *partitionnement d'objet*. Cette distinction se retrouve plus généralement dans le cadre de la visualisation parallèle sous les termes anglophones de *"sort-first"* et *"sort-last"* [Molnar S. *et al.*, 1994] qui font respectivement référence à un *tri précoce* de primitives géométriques en fonction de leur position dans l'espace de l'image et à un *tri tardif* de fragments calculés sur des processeurs différents mais affectant un même pixel de l'image finale.

Partitionnement d'image

Dans le cadre du rendu volumique, le partitionnement d'image consiste simplement à décomposer l'image en sous-parties dont l'intégrale de rendu volumique (Section 2.2.2) est calculée indépendamment sur les différents processeurs. Comme le montre la Figure 2.15, les images calculées sur chaque processeur ont une résolution inférieure à l'image finale et constituent un sous-

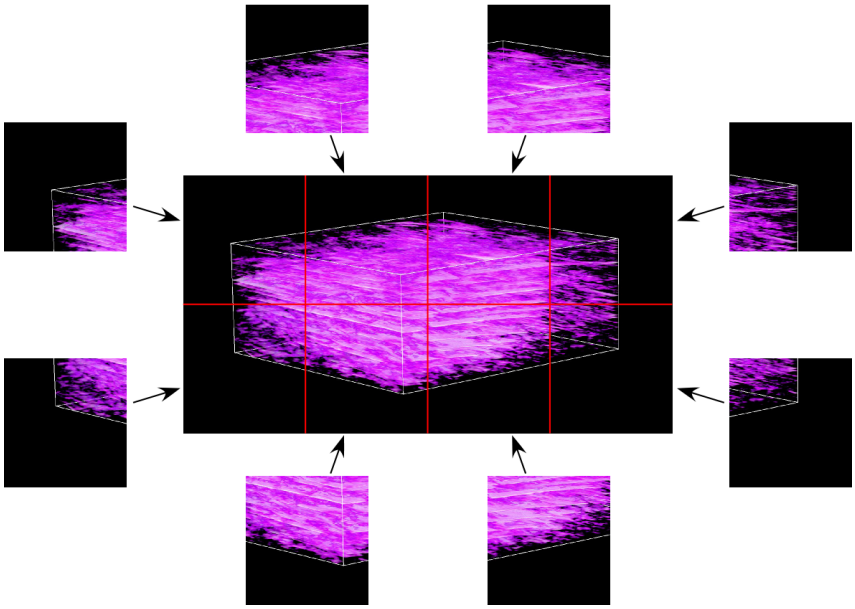

FIG. 2.15 – Principe du *partitionnement d'image* en rendu volumique parallèle. La décomposition de l'image en sous-images de résolution inférieure permet de calculer les intégrales de rendu volumique en parallèle et indépendamment sur les différents processeurs.

ensemble de cette dernière. La reconstitution de l'image finale s'avère donc triviale, consistant simplement à mettre ces dernières côte-à-côte en fonction de leurs positions respectives. Malgré cela, ce type de partitionnement présente l'inconvénient majeur qu'un même bloc de données peut être accédé par plusieurs processeurs différents pour le calcul d'une image seule et plus encore lorsque le point de vue change entre plusieurs images. Ceci suppose donc un partage de données entre processeurs. D'autre part, comme le montre la Figure 2.15, le partitionnement d'image souffre d'un déséquilibre de charge lié à la présence de pixels transparents dans l'image finale. L'utilisation de techniques d'allocation dynamique de sous-tâches [NIEH J. et LEVOY M., 1992, CORRIE B. et MACKERRAS P., 1993, KÖSE C. et CHALMERS A.G., 1997] permet d'y remédier mais reste complexe à mettre en œuvre.

Partitionnement d'objet

Au contraire du partitionnement d'image, le partitionnement d'objet consiste à décomposer le volume en sous-parties dont la contribution à l'ensemble de l'image finale est calculée en parallèle. La Figure 2.16 illustre le fait que les intégrales calculées sur chaque processeur sont seulement des sous-intégrales qu'il convient de composer pour obtenir l'image finale. La validité de cette approche repose sur la propriété d'associativité de l'opérateur de composition (Équation (2.14) ou (2.15) selon le cas). En outre, sa non commutativité impose de tenir compte de l'ordre de superposition des blocs lors de la composition des sous-intégrales pour constituer l'image finale. Cette étape de composition des images produites sur chaque processeur constitue la principale difficulté du partitionnement d'objet et devient critique sur certaines architectures multiprocesseurs, aussi nous aurons l'occasion d'y revenir en Section 2.4.3. En revanche, l'accès concurrent aux données qui nécessite un partage entre processeurs dans le partitionnement d'image est inexistant dans cette approche. De même, l'équilibre de charge découle intrinsèque-

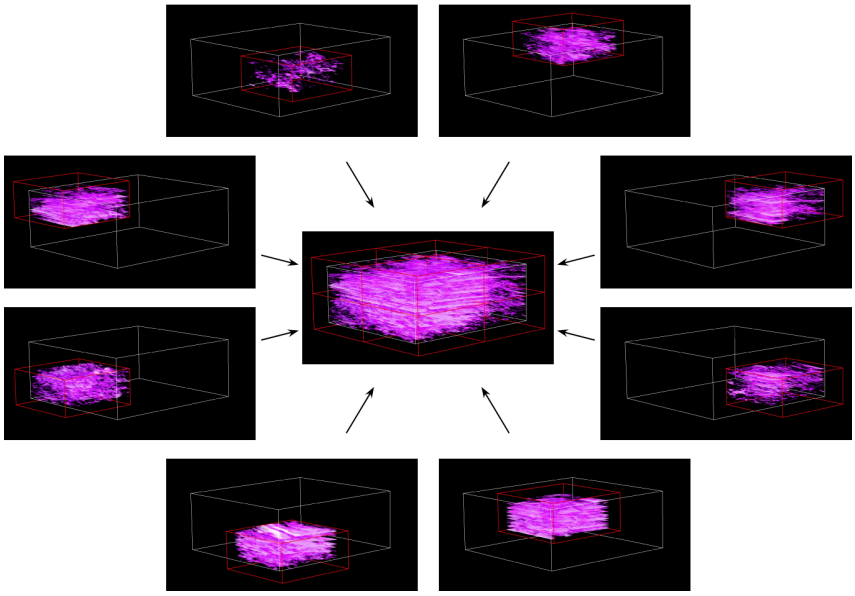

FIG. 2.16 – Principe du *partitionnement d'objet* en rendu volumique parallèle. Des intégrales partielles de rendu volumique correspondant à chaque sous-volume sont calculées sur les différents processeurs et composées pour former l'image finale.

ment du partage équitable du volume de données entre les processeurs.

Au vu des similarités de distinction espace d'image/espace d'objet qu'il existe entre les techniques de rendu volumique et celles de visualisation parallèle, il apparaît judicieux d'en tenir compte lors de la parallélisation. Ainsi, le rendu volumique par lancer de rayon se parallélise naturellement par partitionnement d'image [NIEH J. et LEVOY M., 1992, WHITMAN S., 1993], chaque processeur calculant explicitement l'intégrale de rendu volumique pour un sous-ensemble des pixels de l'image finale. Lors de faibles modifications du point de vue entre plusieurs images, il est possible d'exploiter la cohérence inter-image pour réduire le volume des communications de données entre processeurs [LAW A. et YAGEL R., 1996]. Suivant la même logique, le partitionnement d'objet s'est vu appliqué pour la parallélisation du "splatting" [ELVINS T., 1992, MACHIRAJU R.K. et YAGEL R., 1993, LI P. *et al.*, 1997], du "shear-warp" [AMIN M. *et al.*, 1995, SANO K. *et al.*, 1997] et du rendu volumique par discrétisation implicite à base de textures [PORTER D.H., 2002, KNISS J. *et al.*, 2001b, KIRIHATA Y. *et al.*, 2004, STRENGERT M. *et al.*, 2004], techniques qui fonctionnement par calcul de la contribution de chaque voxel du volume à l'image finale et qui sont donc en parfaite adéquation avec une décomposition du problème dans l'espace de l'objet. Cependant, cette combinaison d'approches de philosophie analogue ne constitue en aucun cas une règle. En effet, le partitionnement d'objet convient parfaitement pour la parallélisation du lancer de rayons [MONTANI C. *et al.*, 1992, SILVA C.T. *et al.*, 1996, PARKER S. *et al.*, 1999]. Dans ce cas, les rayons sont calculés partiellement sur chacun des processeurs et les résultats composés pour obtenir l'image finale. Inversement, le partitionnement d'image est une option de parallélisation pour le "splatting" [HUANG J. *et al.*, 2000] ou le "shear-warp" [AMIN M. *et al.*, 1995, LACROUTE P., 1995, LACROUTE P., 1996, LIN C.F. *et al.*, 2002].

Toutes ces techniques de rendu volumique parallèle nécessitent l'existence de plusieurs processeurs pour traiter l'information en parallèle. Or, les architectures multiprocesseurs ne sont

pas uniques et peuvent différer par leur modèle de programmation. Nous allons donc donner les grandes lignes des principaux systèmes parallèles utilisés pour la visualisation.

2.4.2 Architectures multiprocesseurs et modèles de programmation

Sur une architecture multiprocesseur, le modèle de programmation spécifie d'une part le mode de communication des portions du programme s'exécutant en parallèle sur les différents processeurs, d'autre part les opérations de synchronisation disponibles pour coordonner leurs activités. Il est indépendant de l'architecture physique sous-jacente. Outre le modèle trivial d'applications séquentielles exécutées en parallèle sans communication ni coopération, il existe principalement deux modèles de programmation parallèle pour les architectures multiprocesseurs : le modèle de *mémoire partagée* et le modèle de *mémoire distribuée*.

Mémoire partagée

Dans un modèle de mémoire partagée, l'espace d'adressage est unique et partagé entre tous les processeurs, ce qui permet de communiquer par simple écriture en mémoire. Bien que certaines machines dites à *mémoire globale* disposent effectivement d'une mémoire partagée entre tous les processeurs, ce type de modèle repose généralement pour des raisons de performance et de coût sur une architecture de mémoire physiquement distribuée entre les processeurs. L'accès à l'espace mémoire virtuellement partagé par l'ensemble des processeurs est traduit par le système d'exploitation et les composants matériels en échange de messages sur le réseau d'interconnexion du système. C'est le cas par exemple de la série des SGI Origin [Silicon Graphics Inc., 2002]. De telles architectures possèdent un réseau d'interconnexion à très large bande passante et très faible latence qui, associé à la simplicité du modèle de programmation, en fait des solutions de choix pour la visualisation volumique parallèle.

Mémoire distribuée

Contrairement au modèle de mémoire partagée, le modèle de mémoire distribuée ne fournit aucune zone mémoire d'échange d'information entre les processeurs. Dans ce cas, toute communication s'accompagne invariablement d'un échange de messages explicite entre les processeurs communicant. L'exemple type d'architecture multiprocesseur reposant sur un modèle de mémoire distribuée est le *cluster de PCs*, ou *grappe de PCs*, qui consiste à connecter un ensemble de machines indépendantes, ou *nœuds*, à travers un réseau de communication. La bande passante dans ce type de systèmes est généralement beaucoup plus faible que sur les multiprocesseurs à mémoire partagée. D'autre part, le mode de communication/synchronisation par échange explicite de messages, outre la complexité qu'il introduit, s'accompagne de délais logiciels et matériels qui en accroissent la latence. En revanche, le coût d'acquisition d'un cluster de PCs, qui se compose en fait de composants standards, est très inférieur à celui d'un système multiprocesseur à mémoire partagée.

Malgré l'investissement qu'ils représentent, du fait de l'attractivité de leur modèle de programmation et du différentiel de performance qu'ils induisent, les multiprocesseurs à mémoire partagée constituent une plate-forme de choix pour le développement de solutions de visualisation volumique parallèle, et ce qu'il s'agisse de paralléliser le lancer de rayons [Parker S. *et al.*, 1999], le "splatting" [Machiraju R.K. et Yagel R., 1993], le "shear-warp" [Lacroute P., 1995] ou

même le rendu volumique à base de textures [Kniss J. *et al.*, 2001b]. Cependant, dès le milieu des années 1990 [Becker D.J. *et al.*, 1995] et plus encore aujourd'hui avec le développement exponentiel du matériel graphique standard et des réseaux d'interconnexion entre les machines [Houston M., 2004], les clusters de PCs exposent un rapport qualité-prix extrêmement concurrentiel par rapport aux coûts prohibitifs des solutions à mémoire partagée. Suite à ce constat, de nombreuses solutions de visualisation volumique parallèle sur cluster ont été proposées. L'architecture particulière de mémoire distribuée impose cependant un certain de nombre de contraintes.

Ainsi, comme nous l'avons vu à la section précédente, le partitionnement d'image, bien que naturel dans un contexte de rendu volumique par lancer de rayons, implique un partage de données entre processeurs qui suppose donc l'existence d'un moyen de communication efficace. Pour cette raison, ce type de partitionnement se rencontre communément dans un contexte de mémoire partagée [Nieh J. et Levoy M., 1992, Whitman S., 1993]. Certains auteurs exploitent cependant la cohérence inter-image pour adapter une telle approche aux contraintes d'un modèle de mémoire distribuée [Law A. et Yagel R., 1996]. Des techniques encore plus évoluées reposant sur des implantations logicielles dites de *mémoire partagée distribuée*, ou *DSM* (de l'anglais "Distributed Shared Memory"), mettent en œuvre pour les clusters de PCs un mécanisme proche de la mémoire partagée [Corrie B. et Mackerras P., 1993, Köse C. et Chalmers A.G., 1997, DeMarle D. *et al.*, 2003] et réduisent ainsi les temps d'accès moyens aux données. Nous aurons l'occasion d'y revenir dans le détail au Chapitre 6 dans la mesure où nous avons mis au point une implantation originale de mémoire partagée distribuée pour cluster de PCs destinée à la visualisation et au déplacement en temps réel de sondes volumiques de grande taille dans des volumes de données de très grande taille. Notre système de rendu volumique parallèle repose cependant sur un partitionnement d'objet, une approche beaucoup plus courante sur cluster. En effet, celle-ci n'impliquant pas de partage de données entre processeurs, elle est beaucoup plus adéquate pour une architecture dont le système de communication n'est pas optimal. Cependant, ce type de partitionnement nécessite un algorithme efficace de composition de l'image finale à partir des images calculées sur chaque nœud du cluster. C'est à cette étape de composition que nous allons nous intéresser maintenant.

2.4.3 Partitionnement d'objet sur cluster de PCs

Dans une technique de partitionnement d'objet sur cluster de PCs, qui plus est avec des techniques de rendu volumique modernes qui exploitent la puissance des dernières cartes graphiques, il est important d'avoir conscience que l'étape limitante n'est autre que l'étape de composition de l'image finale. En effet, son implantation à travers le réseau d'interconnexion des nœuds du cluster souffre de bandes passantes limitées et de latences élevées. En d'autres termes, les performances d'un système de rendu volumique parallèle sur cluster sont généralement liées à l'algorithme de composition plus qu'aux performances brutes du moteur de rendu sur chaque nœud, d'où l'intérêt d'y attacher une importance particulière.

La littérature dispose d'un certain nombre de techniques pouvant être classées sous les termes d'*envoi direct* [Hsu W., 1993, Neumann U., 1993], d'*échange binaire* [Ma K.L. *et al.*, 1994], auquel nous préférerons l'expression anglophone *"binary swap"* beaucoup plus courante, et finalement de *pipeline parallèle* [Lee T.Y. *et al.*, 1996]. Des optimisations ont été proposées pour ces techniques de base qui permettent entre autres d'éliminer les pixels transparents dans les communications du "binary swap" par des techniques de compression [Sano K. *et al.*, 2004] et des structures de boîtes englobantes [Ahrens J. et Painter J., 1998, Yang D.L. *et al.*, 2001, Takeuchi A. *et al.*, 2003]. De même, l'algorithme *SLIC* (de l'anglais "Scheduled Linear Image

Compositing") [Stompel A. *et al.*, 2003] est une optimisation de l'envoi direct. Plus que sur les détails d'implantation de ces techniques, il est important d'insister sur la difficulté qu'elles ont, même pour les implantations modernes, à atteindre une vitesse de rendu supérieure à 10 fps[11] sur des volumes de taille pourtant modeste. Ainsi, [Humphreys G. *et al.*, 2002] et leur système Chromium utilisant une technique de "binary swap" rapportent 8 fps pour un volume de 1024^3 voxels et une résolution d'image de 1024^2 sur un cluster de 16 nœuds (bi Intel P4 Xeon 2.4 GHz, ATI Radeon 9800 Pro) connectés en Infiniband 4X. De même, [Strengert M. *et al.*, 2004] rapportent 5 fps pour un volume de $2048^2 \times 1878$ voxels et une résolution d'image de 1024^2 sur un cluster de 16 nœuds (bi AMD 1.6 GHz, NVIDIA GeForce 4 Ti 4600) connectés en Myrinet avec une technique d'envoi direct. Finalement, [Kirihata Y. *et al.*, 2004] obtiennent 14 fps pour un volume de 256^3 voxels et une résolution d'image de 512^2 sur un cluster de 16 nœuds (bi Intel Xeon 1.8 GHz, PNY NVIDIA Quadro FX 3000) connectés en Gigabit Ethernet avec une technique de "binary swap".

Une alternative consiste à recourir à des systèmes de composition implantés au niveau de composants matériels spécialisés. Ainsi, le système Sepia-2 [Lombeyda S. *et al.*, 2001] a servi de support de composition pour [Frank S. et kaufman A., 2004] dans leur système de rendu volumique parallèle sur cluster de PCs équipés avec les cartes VolumePro 1000 de TeraRecon Inc. (Section 2.3.3). La vitesse de rendu est de l'ordre de 6 fps pour une résolution d'image de 1280×1024 avec 9 nœuds du cluster. D'une manière générale, les gains observés sont marginaux en comparaison de la complexité du système et de son coût prohibitif.

Dans le cadre de notre travail, nous avons eu recours à la librairie DViz[12] développée à l'INRIA Lorraine et qui fournit une implantation particulièrement performante du "binary swap" [Cavin X. *et al.*, 2005]. Outre le parallélisme de données avec partitionnement d'objet qui permet le rendu en parallèle de sous-parties du volume sur les nœuds du cluster, chaque nœud implante un parallélisme de contrôle intensif qui permet de superposer les communications, le rendu sur la carte graphique et les calculs de composition, masquant ainsi au maximum les délais introduits lors des communications sur le réseau d'interconnexion des nœuds. Les vitesses de rendu rapportées sont de 45 fps pour une résolution d'image de 1024×768 sur un cluster de 16 nœuds (bi dual-cœur AMD Opteron 275, NVIDIA GeForce 6800 Ultra) avec quatre interfaces Gigabit Ethernet par nœud [Cavin X. et Mion C., 2006]. Nous aurons l'occasion de revenir dans les détails sur l'utilisation que nous faisons de cette librairie dans la Partie III du mémoire concernant l'utilisation de clusters graphiques.

Cette section clôt l'exposé du contexte scientifique de notre travail. Nous disposons maintenant des éléments essentiels à la compréhension du travail exposé dans ce mémoire, à savoir son cadre d'application et la problématique scientifique dans laquelle il s'inscrit. La Partie II concernera la mise en œuvre de techniques particulières pour la visualisation des données sismiques sur une station d'interprétation standard, avant de s'intéresser dans la Partie III à un problème plus précis de visualisation volumique qui nécessite le recours à des clusters graphiques.

[11] de l'anglais "frames per second" (images par seconde)
[12] http ://www.dviz.fr

Deuxième partie

La station d'interprétation sismique

Introduction à la station d'interprétation sismique

Après avoir replacé notre étude dans son environnement applicatif et scientifique nous allons maintenant en aborder les différents aspects en détails. Pour ce faire, nous nous plaçons dans un premier temps dans le contexte de la station d'interprétation sismique moderne telle qu'elle a été présentée en Section 1.3. Les trois chapitres suivants s'inscriront donc dans le cas où l'interprétation prend place sur une station de travail de type PC équipée d'une carte graphique moderne standard (Section 2.3.1) et disposant des outils classiques de l'analyse sismique. Nous avons implanté ces techniques au sein du module VolumeExplorer du logiciel Gocad (Section 1.3). Comme il a été précédemment vu, les techniques existantes d'interprétation du signal sismique reposent principalement sur des outils 2D, avec toutes les limitations que cela comporte pour une bonne compréhension des structures. Notre travail initial a donc concerné la mise en œuvre d'une technique de rendu volumique adaptée aux particularités du signal sismique. Ceci fera l'objet du Chapitre 3. Nous avons ensuite examiné un autre aspect particulièrement important de l'interprétation sismique, à savoir la *multimodalité*. Le terme multimodal fait référence à la coexistence de plusieurs sources d'information dans le processus d'interprétation, par exemple une information d'origine sismique (cube sismique) et une information d'origine structurale (horizons et failles). La capacité à combiner cette information contribue à améliorer le résultat de l'interprétation ainsi qu'à en réduire le temps [Marsh A.J. *et al.*, 2000, Christie M., 2002]. Nous distinguons deux cas de multimodalité successivement abordés au cours du Chapitre 4. Cette distinction porte sur l'espace de définition des différentes sources d'information combinées. Dans le cas où celui-ci est identique, le format de stockage étant le même (grilles régulières), le terme *multi-attribut* peut se substituer au terme multimodal. Cependant, l'information structurale peut prendre différentes formes. En particulier, l'interprétation sismique conduit à l'extraction d'horizons (Section 1.2) et nous considérons les coupes horizon présentées à la Section 1.3 comme une autre forme de visualisation multimodale. Ce point sera donc abordé comme un cas particulier de multimodalité. Finalement, comme il a été souligné en introduction de ce mémoire, du fait des avancées technologiques la taille des volumes sismiques ne cesse de croître, ce qui rend d'autant plus difficile leur manipulation par les logiciels d'interprétation. Nous présenterons donc au Chapitre 5 un système que nous avons mis en place dans le logiciel Gocad prenant en compte les contraintes matérielles liées à la station de travail pour coupler efficacement la visualisation et les calculs dans le contexte de l'interprétation sismique. Ce système sert de support à notre moteur de visualisation volumique et permet donc de regrouper les algorithmes de visualisation présentés aux Chapitres 3 et 4 autour d'un mécanisme commun de gestion de la mémoire. Cependant, un certain nombre de contraintes liées aux impératifs de couplage de la visualisation et des calculs limitent sa portée en termes de visualisation de volumes de très grande taille. Ainsi, le Chapitre 5 servira à la fois de conclusion sur la station d'interprétation sismique et d'introduction aux

notions de base de gestion de la mémoire sur un système simple, en préliminaire d'un système beaucoup plus complexe destiné à la visualisation seule et qui sera abordé dans la Partie III.

Contributions

Nous proposons au Chapitre 3 une analyse fine des principales caractéristiques du signal sismique, notamment en ce qui concerne la distribution spatiale des valeurs d'amplitude dans le volume. Partant de cette analyse, nous identifions les principales raisons de l'échec des techniques de rendu volumique classique dans le contexte de l'interprétation sismique et proposons l'utilisation de techniques modernes plus adaptées ayant fait leurs preuves dans le contexte de l'imagerie médicale. Notre implantation de ces techniques haute qualité dans le module d'interprétation sismique commercial VolumeExplorer du logiciel Gocad nous permet de démontrer le potentiel de la visualisation volumique pour l'interprétation du signal sismique. L'utilisation de ces techniques a fait l'objet de publications dans la communauté informatique [Castanié L. *et al.*, 2005c] et leur impact potentiel sur la démarche d'interprétation en sismique dans la communauté des géosciences [Castanié L. *et al.*, 2005a, Castanié L. *et al.*, 2005b].

Le Chapitre 4 propose tout d'abord une analyse originale du caractère multimodal des données. Cette analyse appliquée aux coupes horizon permet de généraliser la notion à toutes les situations de combinaison d'information indépendamment de l'espace de définition. Pour le cas de données exclusivement volumiques, nous proposons un système générique de visualisation volumique multimodale qui généralise la notion de *calque* des applications de dessin 2D pour combiner les sources avec un maximum de flexibilité. Nous montrons notamment une spécialisation de ce système pour l'extraction et la visualisation en temps réel de surfaces d'isovaleur dites *non polygonales* sur lesquelles est plaquée l'information de couleur issue d'un ou plusieurs volumes supplémentaires. L'originalité du système réside dans la manière qu'il a de combiner les données [Castanié L. *et al.*, 2005c] et dans le potentiel qu'il offre en termes d'interprétation [Castanié L. *et al.*, 2005a, Castanié L. *et al.*, 2005b]. En ce qui concerne les coupes horizon, nous nous inspirons des techniques de visualisation de terrains en niveaux de détail, utilisées notamment dans la conception de jeux vidéo, pour permettre le rendu en temps réel de la géométrie des surfaces. Le recours aux textures 3D disponibles sur le matériel graphique standard nous permet ensuite de projeter interactivement l'information sismique volumique sur la coupe.

Finalement, nous présentons au Chapitre 5 un système de gestion de la mémoire pour manipuler de grandes quantités de données volumiques. L'originalité de ce système est de proposer un cadre d'accès unifié aux données pour la visualisation et les calculs, ce qui suppose plusieurs niveaux d'entrée et la possibilité d'accès en écriture, deux propriétés que n'exposent pas les systèmes classiques destinés à la visualisation seule. D'autre part, il est capable de gérer tous les types de données (8-bit, 16-bit, 32-bit) et dispose de filtres de conversion pour le chargement des textures en mémoire graphique. Finalement, la granularité des accès est de l'ordre du voxel contrairement aux systèmes classiques qui accèdent à des blocs de texture dans leur ensemble. En mémoire centrale, cet accès à grain fin assure la compatibilité de notre système avec les algorithmes existants de calcul qui accèdent aux données en lecture/écriture par voxel. Notre implantation dans le logiciel commercial Gocad nous a permis de l'expérimenter avec succès en situation réelle [Castanié L. *et al.*, 2005c].

Chapitre 3

Visualisation volumique directe du signal sismique

Ce chapitre aborde la visualisation volumique du signal sismique par méthode directe. Les sondes fournies par le logiciel VolumeExplorer (Slicer, Fence, Sections, Skin) sont fondées sur un ensemble de méthodes **indirectes**, à savoir par extraction d'un sous-ensemble du champ scalaire sur des sections plus ou moins arbitraires, dont l'intérêt n'est plus à démontrer dans le cadre de l'interprétation sismique. Cependant, l'absence quasi-systématique de visualisation par méthodes **directes** de rendu volumique (Chapitre 2) dans la démarche des géophysiciens et des géologues ne traduit pas pour autant une désaffection de leur part pour celles-ci. Ceci traduit tout simplement l'inadaptation des techniques classiques de rendu volumique aux propriétés du signal sismique. Après une présentation des principales caractéristiques des données sismiques, nous verrons comment les techniques de rendu volumique modernes peuvent prendre place dans le processus d'interprétation.

3.1 Caractéristiques du signal sismique et de son interprétation

La littérature propose un grand nombre de méthodes de visualisation volumique directe [Levoy M., 1988, Lacroute P. et Levoy M., 1994, Cabral B. *et al.*, 1994]. Toutefois, ces méthodes sont généralement adaptées aux propriétés des données médicales et se révèlent inefficaces dans le contexte de l'exploration pétrolière et des données sismiques. Avant de mettre en œuvre un système de visualisation volumique pour l'interprétation du signal sismique, il convient d'identifier un certain nombre de particularités attachées au signal et à l'utilisation qui en est faite.

3.1.1 Distribution statistique et distribution spatiale

La visualisation volumique s'attache à fournir une représentation particulière d'un volume de données qui permette d'en comprendre la structure, laquelle est intimement liée à la nature du signal volumique étudié. Comme nous l'avons montré au Chapitre 1, le signal sismique correspond à l'enregistrement en continu de la réponse du sous-sol à une impulsion acoustique. Cet enregistrement conduit au cube sismique qui, à chaque emplacement géographique en surface, fait correspondre une trace sismique verticale. Du point de vue de la modélisation physique, le signal sismique le long de chaque trace peut être considéré comme le produit de convolution

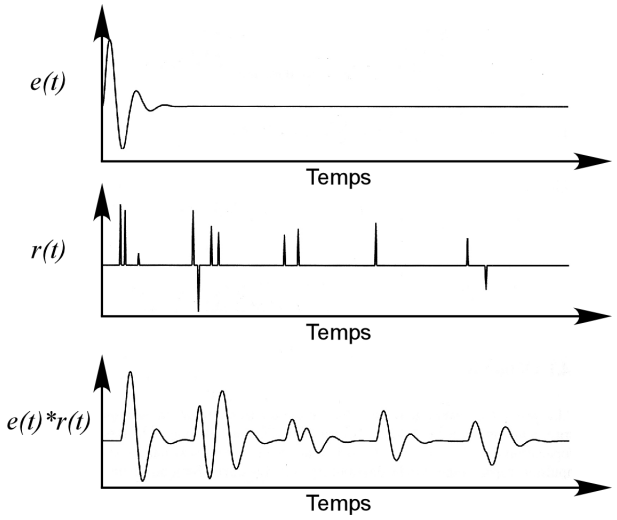

FIG. 3.1 – La réponse acoustique du sous-sol lors de l'acquisition sismique peut être considérée comme le produit de convolution entre une ondelette e correspondant à l'impulsion en surface et un peigne de Dirac r correspondant aux variations des coefficients de réflexion dans les couches géologiques (d'après [LABRUNYE E., 2004]).

entre une ondelette et un peigne de Dirac, correspondant respectivement à l'impulsion en surface et aux variations des coefficients de réflexion dans les couches géologiques (Figure 3.1). Le signal ainsi obtenu est un signal sinusoïdal dont la nature ondulatoire a un fort impact sur la distribution des valeurs d'amplitude dans le cube. Par distribution, nous entendons à la fois distribution statistique et distribution spatiale, lesquelles orientent la démarche de construction de la fonction de transfert en rendu volumique ainsi que le rendu final.

Distribution statistique

D'un point de vue statistique, du fait de la nature ondulatoire du signal sismique, la distribution des valeurs d'amplitude dans le volume est une distribution gaussienne [BROWN A.R., 2004]. Comme le montre la Figure 3.2, KIDD propose dans [KIDD G.D., 1999] une décomposition de cette distribution en six zones : trois zones d'amplitudes négatives et trois zones d'amplitudes positives. Les extremums de la distribution correspondent aux zones 1 (Z1-/Z1+), les valeurs intermédiaires aux zones 2 (Z2-/Z2+) et enfin les valeurs moyennes aux zones 3 (Z3-/Z3+). Comme nous le verrons à la section suivante, cette décomposition est à la base de son système *ZS* (de l'anglais "Zone System") d'édition de fonction de transfert.

Distribution spatiale

Sur le plan spatial, la conséquence de la nature ondulatoire du signal sismique est une organisation lenticulaire sous la forme de couches concentriques autour des valeurs extrêmes. Comme l'illustre la Figure 3.3, les cœurs lenticulaires présentent une forme allongée (aplatie en 3D). Dans ce cas précis, ils correspondent aux zones Z1, mais sont également souvent classés dans les zones Z2. Entre un cœur Z1- et un cœur Z1+, les couches concentriques balaient l'ensemble des zones Z2 et Z3 dans l'ordre Z2-/Z3-/Z3+/Z2+. Cette organisation lenticulaire aura des conséquences importantes dans les sections suivantes pour comprendre l'échec des méthodes classiques

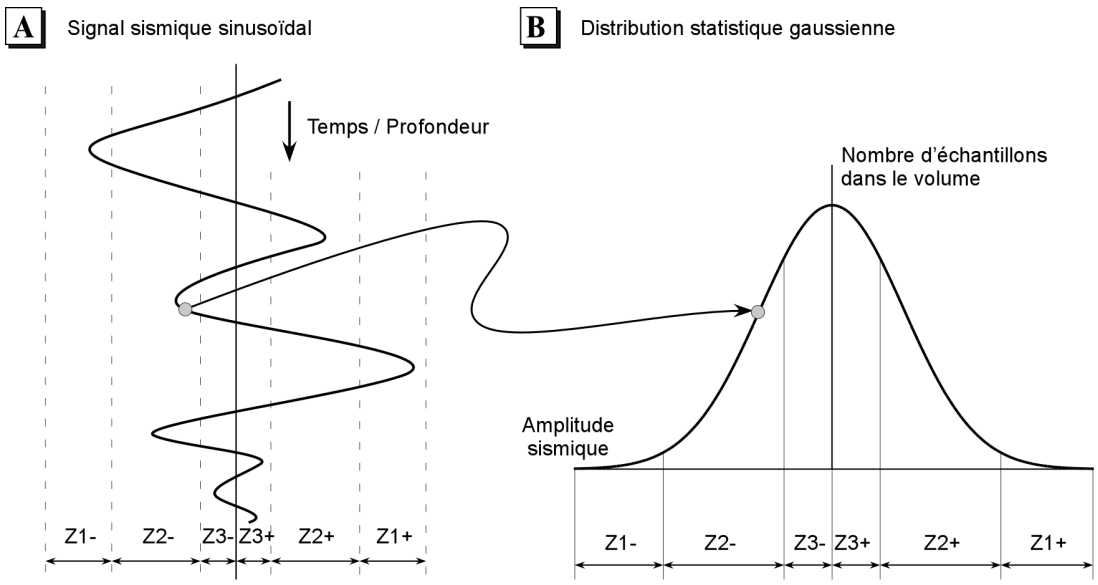

FIG. 3.2 – Du fait de la nature ondulatoire du signal sismique (A) les valeurs d'amplitude ont une distribution gaussienne (B). Une décomposition de cette distribution en zones permet de distinguer trois groupes d'amplitudes négatives ou positives : les amplitudes extrêmes Z1-/Z1+, intermédiaires Z2-/Z2+ et moyennes Z3-/Z3+ (modifié d'après [KIDD G.D., 1999]).

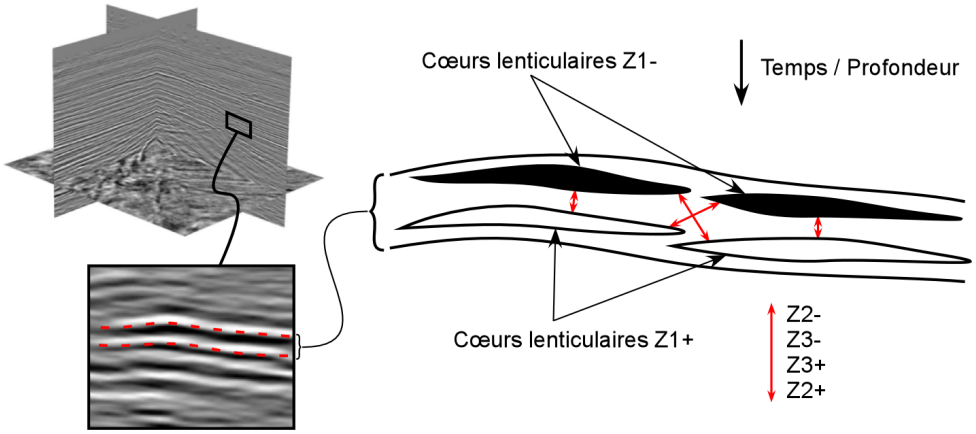

FIG. 3.3 – Organisation lenticulaire des amplitudes sismiques en couches concentriques autour des valeurs extrêmes. Les cœurs lenticulaires correspondent ici aux zones Z1 et les couches concentriques balaient l'ensemble des zones Z2 et Z3.

de rendu volumique avec les données sismiques et nous orienter vers la définition de techniques plus adaptées.

3.1.2 Interprétation structurale du signal sismique

Comme nous l'avons vu en Section 1.2.2, il existe différentes formes d'analyse de l'image sismique. Le rendu volumique ne présente que peu d'intérêt pour l'analyse stratigraphique, tout au moins concernant la partie effectuée à échelle régionale pour laquelle les outils de visualisation 2D classiques sont déjà bien adaptés. Pour ce qui est des analyses structurale et pétrophysique, elles sont étroitement liées aux distributions statistique et spatiale présentées à la section précédente. En effet, selon Brown chaque zone d'amplitudes sismiques de la distribution statistique est susceptible de fournir une information différente [Brown A.R., 2004]. Ainsi, il est admis que les valeurs extrêmes d'amplitude des zones Z1 correspondent à des anomalies locales et donc à des cœurs lenticulaires de faible extension, lesquels sont souvent interprétés comme étant des "bright spots". Au contraire, l'information structurale est contenue dans ce que Brown appelle les *amplitudes modérées* à cheval entre les zones Z2 et Z3, formant selon les cas des cœurs lenticulaires ou des enveloppes concentriques autour de cœurs Z1/Z2. Finalement, les valeurs les plus proches de la moyenne, à la limite entre les zones Z3- et Z3+, sont généralement sujettes au bruit et présentent peu d'intérêt dans l'interprétation.

Rendu volumique du signal sismique

Le rendu volumique se comporte bien dans les plages d'amplitudes Z1. Dans la mesure où elles correspondent à des événements ponctuels, il suffit de choisir une fonction de transfert qui associe une opacité maximale aux amplitudes de la zone Z1- ou Z1+ et rend le reste transparent de façon à mettre rapidement en évidence la localisation de ces événements. Cependant, dans le cadre de l'interprétation structurale, le travail porte majoritairement sur l'observation des plages d'amplitudes Z2 et Z3 beaucoup plus "peuplées". L'ajustement de l'opacité dans ces zones est par conséquent beaucoup plus complexe et les images produites par le rendu volumique souffrent généralement de phénomènes d'occlusion. C'est dans cette optique que Kidd a mis au point son système ZS [Kidd G.D., 1999]. Afin de limiter les phénomènes d'occlusion, une première étape consiste à isoler un objectif observé à grande échelle, en ne conservant qu'une fenêtre temps/profondeur verticale calée sur un événement sismique marqué tel qu'un horizon par exemple. Puis, après avoir clairement identifié les différentes zones Z1/Z2/Z3 dans cette fenêtre, ce système permet une sélection fine des couleurs et des niveaux d'opacité de manière à contraster les événements distincts. Kidd parvient ainsi à mettre très clairement en évidence les différentes unités dans des formations fluviatiles ou encore à observer précisément l'impact d'un réseau de failles sur un groupe d'horizons. Cependant, cette approche entièrement manuelle repose sur une connaissance approfondie des propriétés de contraste entre couleurs ainsi que de leur comportement en fonction du niveau d'opacité, ce qui demande un long apprentissage. En effet, un tel jeu de contraste repose sur l'affichage d'un large intervalle d'amplitudes sismiques et de très légères modifications de la courbe d'opacité ont de grandes conséquences sur le rendu final. Ceci explique également la nécessité d'isoler préalablement une sous-partie du volume.

Une méthode semi-automatique combinant segmentation et rendu a également été proposée [Gerhardt A. *et al.*, 1999]. Dans cette méthode, un algorithme de segmentation génère une probabilité d'appartenance à la région d'intérêt dans un prétraitement en fonction de paramètres stockés ou évalués au niveau de chaque voxel (amplitude, gradient, ...). Puis, une implantation

du "splatting" (Section 2.2.2) permet de sélectionner le niveau d'opacité à partir d'une fonction de transfert qui tient compte à la fois de valeurs spécifiées par l'utilisateur et de la probabilité d'appartenance qui joue ainsi le rôle de masque. Les images produites ne sont cependant pas dépourvues de phénomènes d'occlusion. D'autre part, malgré l'assistance fournie à l'utilisateur pour la sélection des valeurs d'opacité, cette approche lui laisse la lourde tâche de définir manuellement la palette de couleurs. Une bonne connaissance des propriétés de contraste entre couleurs reste donc nécessaire pour mettre en évidence les événements sismiques intéressants.

Fonctions de transfert pour l'interprétation structurale

Pour favoriser l'adoption de techniques de rendu volumique dans le processus d'interprétation sismique, il est nécessaire de faciliter le travail d'édition de la fonction de transfert. Dans la nature, un horizon est caractérisé par la continuité latérale du contraste d'impédance acoustique entre deux couches géologiques. Ainsi, son expression sur l'image sismique se traduit par la continuité latérale de l'amplitude sur une grande série de traces. C'est donc cette continuité que l'analyse structurale de l'image sismique est chargée d'étudier, de même que ses interruptions qui traduisent généralement la présence de failles.

Comme nous l'avons dit précédemment, les amplitudes qui correspondent à des contrastes d'impédance acoustique traduisant des phénomènes structuraux se situent généralement à la frontière entre les zones Z2 et Z3. Ainsi, dans le cadre de l'interprétation structurale du signal sismique, l'édition de la fonction de transfert du rendu volumique nécessite un maximum de flexibilité dans cet intervalle de valeurs. La tendance naturelle conduit à éditer des pics d'opacité dans les amplitudes modérées, laissant ainsi la grande majorité des valeurs transparentes et ne gardant que quelques valeurs opaques. Ceci revient en d'autres termes à visualiser des surfaces d'isovaleur de l'amplitude sismique par rendu volumique. Du point de vue de la distribution spatiale, ces surfaces correspondent soit à des cœurs lenticulaires Z2 aplatis et d'extension particulièrement grande, soit à des enveloppes concentriques très étendues englobant un ou plusieurs cœurs Z1/Z2 (Figure 3.3).

Dans le cas particulier de cœurs lenticulaires, ceux-ci sont encadrés par des valeurs de la zone Z3, ce qui signifie qu'un pic d'amplitude a été atteint sur la trace sismique, point où la dérivée du signal s'annule. Dès lors, Gerhardt *et al.* proposent une sélection plus fine des valeurs d'opacité en fonction non seulement de la valeur d'amplitude sismique mais aussi de sa dérivée verticale [Gerhardt A. *et al.*, 2001, Gerhardt A. *et al.*, 2002]. Pour cela, ils utilisent une fonction de transfert à deux dimensions qui tient compte des deux paramètres. Cette technique a longtemps été éprouvée dans le domaine médical où elle permet notamment de visualiser l'interface entre deux organes [Levoy M., 1988]. Elle a également été portée sur des architectures modernes bénéficiant ainsi de l'accélération matérielle et d'outils d'édition de la fonction de transfert particulièrement interactifs [Kniss J. *et al.*, 2001a, Kniss J. *et al.*, 2002a]. Par l'utilisation de cette fonction de transfert bidimensionnelle, leur approche permet de filtrer tous les cas où la surface d'isovaleur correspond à une enveloppe et non à un cœur lenticulaire. Leur but est de réduire les phénomènes d'occlusion. Cependant, un inconvénient majeur de cette méthode est l'approximation ainsi réalisée, risquant d'ignorer des éléments d'information importants pour le processus d'interprétation.

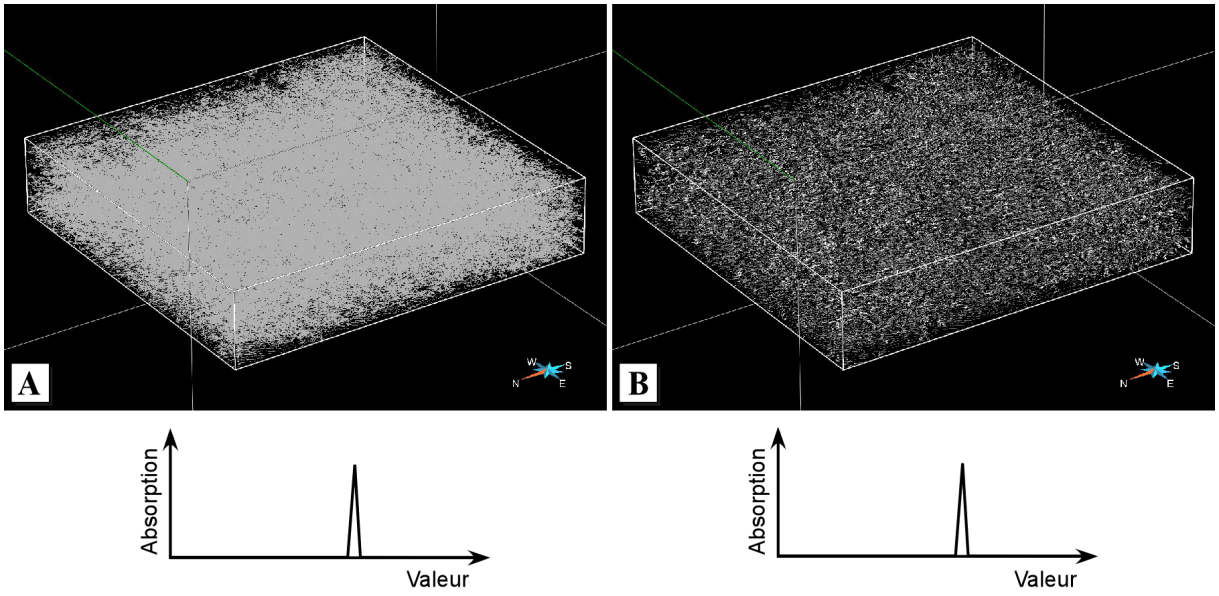

FIG. 3.4 – Rendu volumique par méthode implicite à base de textures 3D avec post-classification sur des données sismiques sans (A) et avec (B) application d'un modèle d'illumination locale (Section 3.2.1). Les courbes d'opacité sont données. Malgré un sur-échantillonnage (deux plans par voxel en moyenne), l'introduction de hautes fréquences dans la fonction de transfert (pic d'opacité) perturbe le comportement du rendu volumique classique à base de post-classification.

3.1.3 Limitations du rendu volumique classique

L'approche de GERHARDT *et al.* vue à la section précédente repose sur une technique classique de rendu volumique par méthode implicite à base de textures 3D (Section 2.3.2) avec post-classification (Section 2.2.2). Or, cette dernière supporte très mal l'introduction de hautes fréquences dans la fonction de transfert. La Figure 3.4 illustre le phénomène sur des données sismiques. La fonction de transfert utilisée est constituée d'un simple pic d'opacité dans les amplitudes modérées (les autres valeurs sont laissées transparentes) et la distance entre les plans d'échantillonnage est égale à la taille d'un demi-voxel. L'image ainsi produite n'est clairement pas interprétable. La compréhension du phénomène est plus aisée sur un objet familier. Ainsi, la Figure 3.5-A montre le résultat obtenu pour une fonction de transfert similaire sur des données de scanner d'une dent, laquelle est censée représenter l'enveloppe superficielle de la dent. La distance entre les plans d'échantillonnage dans ce cas est égale à la taille d'un voxel et les isocontours opaques observés correspondent à l'intersection entre l'enveloppe superficielle de la dent et ces plans. Selon la théorie du traitement du signal, du fait des très hautes fréquences introduites dans la fonction de transfert par l'utilisation d'un pic d'opacité, la fréquence de Nyquist de reconstruction correcte de l'intégrale de rendu volumique est extrêmement élevée. En effet, celle-ci ne dépend pas uniquement de la fréquence de Nyquist du champ scalaire, mais également de celle de la fonction de transfert [SCHULZE J.P. *et al.*, 2003, KRAUS M., 2003]. Ainsi, nous nous trouvons face à une situation de sous-échantillonnage de l'intégrale de rendu volumique.

Une approche naïve consisterait à réduire la distance entre les plans d'échantillonnage. Cependant la puissance de rasterisation des cartes graphiques ainsi que le flux de fragments qui peuvent être traités en temps réel par son processeur de fragment sont limités, ce qui réduit considérablement les performances lorsque le nombre de plans augmente de manière significative. Une autre approche consiste à réduire les fréquences de la fonction de transfert en adoptant

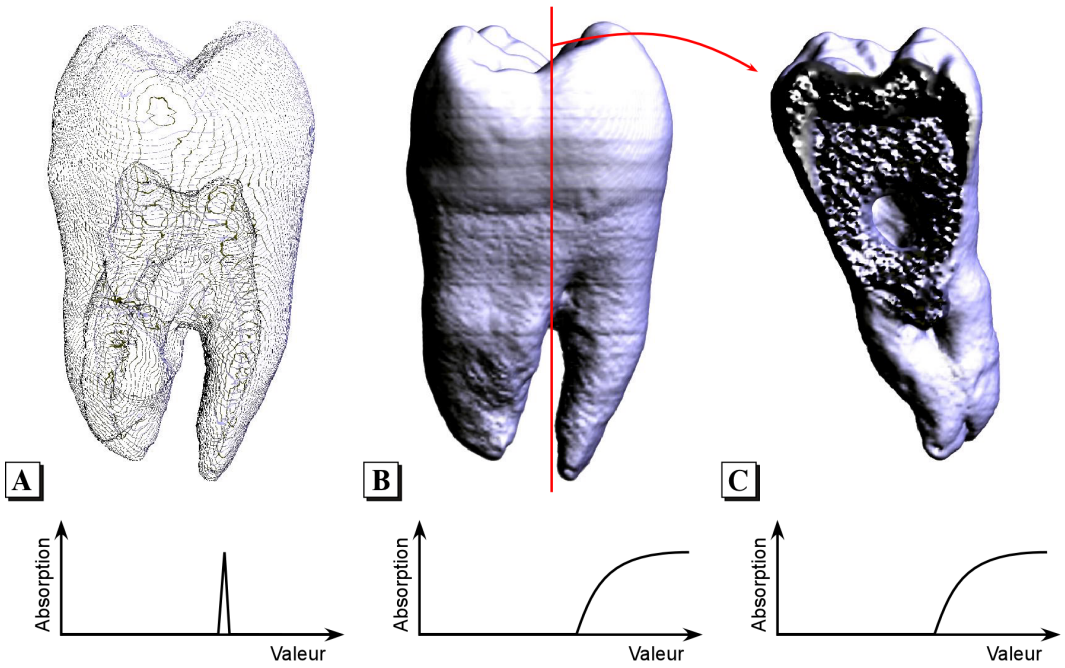

FIG. 3.5 – Rendu volumique par méthode implicite à base de textures 3D sur des données de scanner d'une dent avec post-classification. (A) Un pic d'opacité dans la fonction de transfert introduit de hautes fréquences qui ne peuvent être correctement capturées. (B) Malgré la persistence d'artefacts, la suppression des hautes fréquences dans la courbe d'opacité conduit à un bien meilleur résultat. (C) Cependant, une coupe transversale montre que le rendu ne porte plus sur la seule enveloppe superficielle de la dent. Dans tous les cas, un modèle d'illumination locale est appliqué (Section 3.2.1).

des courbes d'opacité plus douces, comme le montre la Figure 3.5-B. Cette approche conduit à des images plus exploitables, bien que des artefacts de sous-échantillonnage soient toujours visibles. Cependant, comme le montre clairement la coupe transversale sur la Figure 3.5-C, le rendu dans ce cas ne porte plus sur la seule enveloppe superficielle de la dent. Par ailleurs, considérant la nécessaire flexibilité dans les plages d'amplitudes modérées évoquée précédemment, il est évident que cette contrainte de réduction des hautes fréquences dans la fonction de transfert condamne l'utilisation du rendu volumique classique à base de post-classification dans le cadre de l'interprétation sismique.

Nous nous sommes donc intéressés aux techniques modernes de rendu volumique introduites pour les besoins de l'imagerie médicale et capables de produire un résultat correct dans de telles conditions. Ces techniques seront développées à la section suivante tandis que la Section 3.3 donnera les détails de notre implantation dans le logiciel VolumeExplorer pour le cas de la visualisation du signal sismique.

3.2 Notre méthode de rendu volumique adaptée aux propriétés du signal sismique

Nous venons de voir quelles sont les principales caractéristiques du signal sismique ainsi que les limitations du rendu volumique classique dans le contexte de l'interprétation sismique. Cette section a pour but de présenter une méthode de rendu volumique adaptée aux propriétés du signal sismique. Elle repose sur l'utilisation de calculs d'illumination ainsi que sur des techniques de rendu volumique haute qualité initialement introduites pour les besoins de l'imagerie médicale.

3.2.1 Modèle d'illumination locale en rendu volumique

La motivation principale qui a conduit à l'introduction de modèles d'illumination en informatique graphique est la recherche constante de réalisme pour tendre vers ce qui est appelé le rendu *photo-réaliste*. En parallèle, cette quête a inspiré le développement de modèles d'illumination pour le rendu volumique, lesquels permettent de renforcer l'image produite par le rendu volumique dans son rôle de vecteur d'information.

Modèles d'illumination en informatique graphique

Par définition, les images produites par le pipeline graphique présenté en Section 2.3.1 sont des projections dans l'espace 2D de l'écran d'un modèle virtuel 3D. Du point de vue de l'observateur, cette représentation 2D s'accompagne inévitablement d'une altération de l'information de profondeur. L'un des moyens de réduire ce phénomène est d'introduire des modèles d'illumination qui simulent le comportement des objets visualisés en présence de sources lumineuses virtuelles. Les ombres ainsi simulées recréent une partie de l'information de profondeur, en fournissant des "indices visuels" que le cerveau va interpréter. Considérons un objet représenté par un maillage polygonal, à savoir par un ensemble de facettes planaires. L'éclairage d'une facette dépend de son orientation ainsi que des positions des sources lumineuses et de l'observateur. Le résultat des calculs d'illumination sur chaque primitive graphique dépend de deux éléments distincts du modèle : l'*éclairage* (*"lighting"* en anglais) et l'*ombrage* (*"shading"* en anglais) [Foley J.D. *et al.*, 1996], ce qui explique l'utilisation de l'expression "lighting & shading" pour faire référence à l'application d'un modèle d'illumination. L'éclairage concerne le calcul des interactions lumineuses à proprement parler et définit le caractère local ou global du modèle selon qu'il simule seulement les interactions primaires entre sources et objets (directes) ou bien l'ensemble des réflexions lumineuses (directes, indirectes, ombres portées, caustiques, ...). L'ombrage quant à lui porte sur la manière dont le modèle est appliqué sur les primitives. L'ombrage dit de Gouraud consiste à calculer l'éclairage pour chaque sommet et à interpoler le résultat sur les primitives [Gouraud H., 1971] tandis que l'approche dite de Phong, ou *illumination par pixel*, interpole les normales sur les primitives et calcule l'éclairage pour chaque fragment [Phong B.T., 1975]. Le terme "par pixel" est un abus de langage dans la mesure où le calcul a effectivement lieu au niveau du fragment, plusieurs fragments pouvant correspondre à un même pixel (Section 2.3.1).

Modèle optique d'émission-absorption-réflexion en rendu volumique

Bien qu'initialement introduits pour donner plus de réalisme à la représentation graphique d'objets surfaciques, les calculs d'illumination peuvent également intervenir dans les modèles

de comportement optique des particules pour le rendu volumique. Ainsi, le modèle d'émission-absorption présenté en Section 2.2.2 peut être étendu à la prise en compte des réflexions lumineuses, ce qui conduit au *modèle d'émission-absorption-réflexion* dont les premières traces apparaissent dans les années 1980 [Tuy H. et Tuy L., 1984] et qui définit l'intégrale de rendu volumique comme suit :

$$\left| \begin{array}{l} I \;=\; \displaystyle\int_0^D E(t) \;\cdot\; e^{-A(t)}\; dt \\ \\ \text{avec :} \left\{ \begin{array}{rcl} E(t) & = & e\big(\mathbf{x}(t)\big) + r\big(\mathbf{x}(t)\big) \\ \\ A(t) & = & \displaystyle\int_0^t a\big(\mathbf{x}(t')\big)\; dt' \end{array} \right. \end{array} \right. \tag{3.1}$$

où les notations de l'Équation (2.4) sont inchangées, à savoir que D est la distance au point où le rayon quitte le volume, $e(\mathbf{x})$ et $a(\mathbf{x})$ respectivement l'intensité lumineuse émise et absorbée au point $\mathbf{x}$, avec le terme $r(\mathbf{x})$ supplémentaire qui prend en compte les phénomènes de réflexion liés aux sources lumineuses. Ce terme se calcule indépendamment pour chaque source lumineuse comme suit :

$$r\big(\mathbf{x}(t)\big) \;=\; \sum_1^{n_l} r_i\big(\mathbf{x}(t)\big) \tag{3.2}$$

où n_l est le nombre de sources lumineuses et $r_i(\mathbf{x})$ la part d'intensité réfléchie issue de la source i. Par la suite, nous considérerons le cas simple et répandu d'une seule source lumineuse, l'extension au cas de plusieurs sources étant triviale.

Modèle d'illumination locale de Phong

Étant donné la nature de l'objet d'étude, la notion de réalisme en rendu volumique, qui plus est sur des données géologiques, n'a évidemment pas de sens. L'utilisation d'un modèle d'illumination dans ce cas permet simplement une meilleure compréhension de l'orientation dans l'espace des structures observées. Les calculs d'illumination interviennent au niveau de chaque particule et la normale aux surfaces utilisée dans le pipeline graphique standard est remplacée ici par le gradient du champ scalaire. D'autre part, le modèle appliqué est généralement un modèle local [Lichtenbelt B. *et al.*, 1998], dit *modèle d'illumination de Phong* à ne pas confondre avec l'ombrage de Phong. Dans ce modèle, les effets de la lumière réelle sont simulés par trois composantes distinctes :

$$r(\mathbf{x}) \;=\; r_a(\mathbf{x}) \;+\; r_d(\mathbf{x}) \;+\; r_s(\mathbf{x}) \tag{3.3}$$

où $r_a(\mathbf{x})$, $r_d(\mathbf{x})$ et $r_s(\mathbf{x})$ sont respectivement les composantes *ambiante*, *diffuse* et *spéculaire*, chacune ayant un sens différent :

– **Lumière ambiante :** Le terme ambiant est un terme d'illumination constant qui s'exprime comme suit :

$$r_a(\mathbf{x}) \;=\; i_a \tag{3.4}$$

où i_a est le niveau de lumière ambiante défini globalement dans l'environnement virtuel. Ce terme correspond au rayonnement lumineux de très faible intensité sans direction préférentielle. C'est lui qui rend un objet visible dans une pièce sombre.

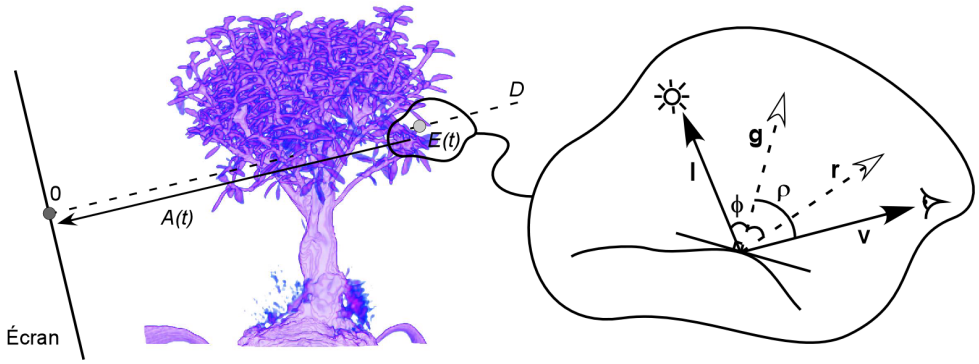

FIG. 3.6 – Géométrie des interactions lumineuses dans un modèle d'illumination locale.

– **Lumière diffuse :** Le terme diffus simule les *réflexions de Lambert*, égales dans toutes les directions de l'espace et ne dépendant donc que de la position de la source de lumière par rapport à l'objet. Il s'exprime comme suit :

$$r_d(\mathbf{x}) \;=\; e(\mathbf{x})\; c_l\; \cos(\phi) \;=\; e(\mathbf{x})\; c_l\; (\mathbf{l}\cdot\mathbf{g}) \tag{3.5}$$

où c_l est l'intensité lumineuse reçue par la particule depuis la source et ϕ est l'angle entre la direction du rayon lumineux incident $\mathbf{l}$ et le gradient local du champ scalaire $\mathbf{g}$ (Figure 3.6). À noter la présence du terme d'émission de la particule $e(\mathbf{x})$ (Équations (2.4) et (3.1)) introduit à la Section 2.2.2 et responsable de la couleur mate de l'objet.

– **Lumière spéculaire :** Le terme spéculaire simule les effets de brillance sur un objet, lesquels dépendent à la fois de la position de la source par rapport à l'objet et de la position de l'observateur. Son expression est la suivante :

$$r_s(\mathbf{x}) \;=\; e(\mathbf{x})\; c_l\; \cos(\rho-\phi)^n \;=\; e(\mathbf{x})\; c_l\; (\mathbf{r}\cdot\mathbf{v})^n \tag{3.6}$$

où ρ est l'angle entre la direction d'observation $\mathbf{v}$ et le gradient local du champ scalaire $\mathbf{g}$, $(\rho-\phi)$ représentant donc l'angle entre la direction d'observation et la direction du rayon lumineux $\mathbf{r}$ réfléchi selon la loi de Snell-Descartes (Figure 3.6). L'exposant n, qui est appelé *spécularité* de l'objet et traduit son caractère plus ou moins brillant, a un impact direct sur la taille des "tâches" spéculaires. Ce terme peut s'avérer lourd à calculer dans la mesure où le vecteur $\mathbf{r}$, qui dépend de la direction du gradient local $\mathbf{g}$, doit être évalué en chaque point. Le *modèle d'illumination de Blinn-Phong* [BLINN J.F., 1977] propose de remplacer le produit scalaire $(\mathbf{r}\cdot\mathbf{v})$ comme suit :

$$r_s(\mathbf{x}) \;=\; e(\mathbf{x})\; c_l\; (\mathbf{r}\cdot\mathbf{v})^n \;\approx\; e(\mathbf{x})\; c_l\; (\mathbf{h}\cdot\mathbf{g})^n \tag{3.7}$$

où $\mathbf{h}$ est le vecteur bissecteur de l'angle formé par les vecteurs $\mathbf{l}$ et $\mathbf{v}$.

Dans la pratique, le modèle d'illumination est calculé dans l'espace de couleurs *RGB* de manière indépendante sur chaque composante. D'autre part, le gradient est stocké par voxel et interpolé au niveau de chaque particule pour les calculs d'illumination, ce qui s'apparente à l'ombrage de Phong pour l'illumination par pixel des surfaces polygonales [PHONG B.T., 1975]. De nombreuses implantations ont été proposées qui tirent profit de l'accélération par le matériel graphique [DACHILLE F. *et al.*, 1998, MEISSNER M. *et al.*, 1999, REZK-SALAMA C. *et al.*, 2004]. La Figure 3.7 montre l'effet de l'application d'un modèle d'illumination de Phong sur des données de scanner d'un crâne humain. Sur l'image de la Figure 3.7-A produite sans calcul d'illumination,

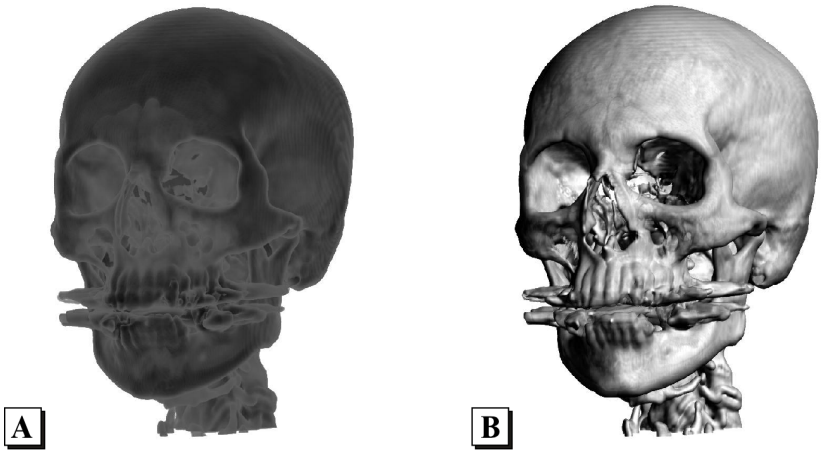

FIG. 3.7 – Rendu volumique sur des données de scanner d'un crâne humain sans (A) et avec (B) application d'un modèle d'illumination locale. La fonction de transfert est identique dans les deux cas.

il est difficile de distinguer clairement les structures, notamment au niveau de la cavité nasale. Au contraire, les effets d'éclairage sur l'image de la Figure 3.7-B permettent une bonne compréhension de la structure du champ scalaire. Dans le cas particulier du rendu volumique de la sismique, certains auteurs proposent de remplacer le gradient du champ scalaire (amplitude sismique) dans les calculs d'illumination par le gradient de l'attribut de phase instantanée du signal (Section 4.1.1) qui reflète mieux l'orientation des horizons sismiques [SILVA P.M. *et al.*, 2003]. Ceci rejoint l'idée que l'orientation des structures d'intérêt en sismique ne suit pas forcément le gradient du champ scalaire observé, idée ayant conduit par exemple au calcul de l'*attribut de relief* [BARNES A.E., 2002, BARNES A.E., 2003].

Ombres volumiques

Dans le modèle d'émission-absorption-réflexion que nous venons de présenter, le terme c_l traduit l'intensité lumineuse reçue par la particule depuis la source. Sans précisions supplémentaires, il est possible de considérer cette intensité constante pour l'ensemble des particules du volume. Cependant, il est intéressant de citer les approches plus évoluées qui reproduisent l'effet des ombres portées en tenant compte de l'atténuation de la lumière entre la source et chaque particule, formant ainsi des *ombres volumiques* [KAUFMAN A. et MUELLER K., 2005]. c_l dépend alors de la position $\mathbf{x}$ de la particule comme suit :

$$\left| \begin{array}{l} c_l\big(\mathbf{x}(t)\big) \;=\; C_l \;\cdot\; e^{-A_l(t)} \\ \\ \text{avec : } A_l(t) \;=\; \displaystyle\int_t^T a\big(\mathbf{x}(t')\big)\; dt' \end{array} \right. \tag{3.8}$$

où C_l est l'intensité lumineuse émise par la source, T est la distance entre la particule en $\mathbf{x}$ et la source, l'intégration de t à T se faisant le long du segment qui les relie, et $a(\mathbf{x})$ est l'intensité lumineuse absorbée au point $\mathbf{x}$ (Équation (2.4), Section 2.2.2). Par analogie avec l'Équation (2.4), C_l représente une émission ponctuelle au niveau de la source et $A_l(t)$ son atténuation par absorption liée aux particules rencontrées entre le point d'émission et le point $\mathbf{x}$. Plusieurs implantations du "splatting" qui génèrent des ombres volumiques par cette approche ont été proposées

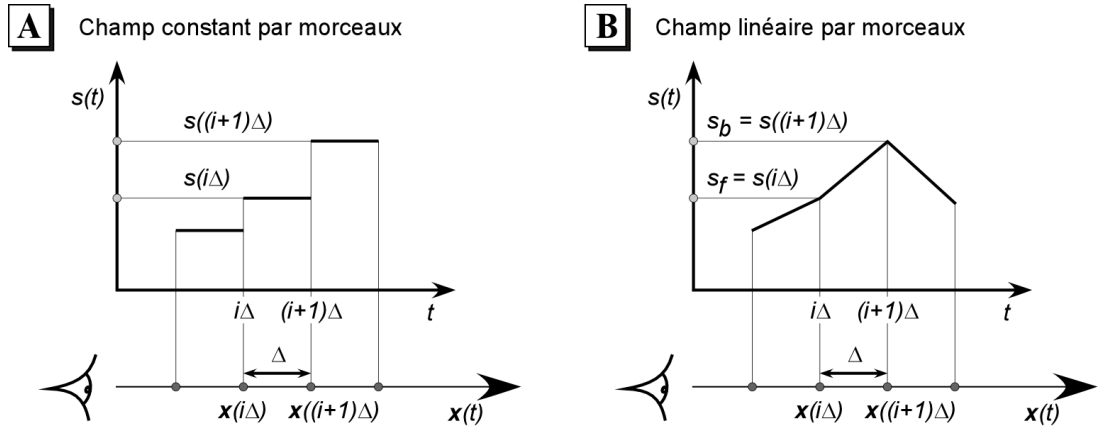

FIG. 3.8 – Approximations champ constant (A) ou linéaire (B) par morceaux lors de l'intégration numérique de l'intégrale de rendu volumique (modifié d'après [ENGEL K. *et al.*, 2001]).

[NULKAR M. et MUELLER K., 2001, ZHANG C. et CRAWFIS R., 2002]. Le surplus de calculs introduit réduit cependant significativement les performances. Une autre technique bénéficiant de l'accélération par le matériel graphique a vu le jour [BEHRENS U. et RATERING R., 1998]. Elle souffre d'un coût mémoire important, deux fois la taille du volume devant être stockée sous forme de textures en mémoire graphique. Des améliorations ont cependant permis de réduire ce coût à une seule texture 2D [KNISS J. *et al.*, 2002b, KNISS J. *et al.*, 2003a], rendant ainsi possible le calcul d'ombres volumiques en temps réel sans coût mémoire supplémentaire.

Dans le cadre de l'interprétation sismique, le modèle d'émission-absorption-réflexion avec illumination locale sans atténuation de l'intensité lumineuse entre source et particule est amplement suffisant. Aussi, par la suite nous considérerons qu'un tel modèle est appliqué dès lors qu'un calcul d'illumination interviendra. D'autre part, dans la mesure où notre but est de visualiser le volume à l'aide de fonctions de transfert contenant des pics d'opacité dans les amplitudes modérées (Section 3.1.2), ce qui revient à visualiser des surfaces d'isovaleur en rendu volumique, nos calculs d'illumination seront basés sur le gradient du champ scalaire. La Section 3.3.1 montrera comment la combinaison d'un programme de sommet et d'un programme de fragment nous a permis d'intégrer un tel modèle d'illumination dans notre moteur de rendu volumique pour la visualisation des données en sismique.

3.2.2 Rendu volumique préintégré

Comme nous l'avons expliqué en Section 3.1.3, le rendu volumique classique à base de post-classification présente un certain nombre de limitations qui réduisent le cadre pratique de son utilisation à des fonctions de transfert dépourvues de hautes fréquences. Cette contrainte en fait un candidat mal adapté pour la visualisation des données sismiques dans le cadre de l'interprétation structurale. En effet, le niveau de flexibilité requis dans la plage des amplitudes modérées conduit naturellement à utiliser des pics d'opacité dans la fonction de transfert, ce qui introduit de très hautes fréquences. Or, la raison de cette sensibilité aux hautes fréquences réside dans l'approximation qui est faite de l'émission et de l'opacité dans les segments lors de la résolution numérique de l'intégrale de rendu volumique, approximation de l'Équation (2.12)

que nous rappelons ici :

$$\left| \begin{array}{rcl} C_i & \approx & c\big(s(i\Delta)\big)\ \Delta \\ \\ \alpha_i & \approx & \tau\big(s(i\Delta)\big)\ \Delta \end{array} \right.$$

où C_i et α_i sont respectivement l'émission et l'opacité du segment i, $c(s)$ et $\tau(s)$ les fonctions de transfert de l'émission propre et de l'absorption, et Δ la longueur des segments d'intégration.

Cette approximation fait l'hypothèse d'un champ scalaire constant par morceaux, la valeur le long de chaque segment restant égale à la valeur en entrée (Figure 3.8-A). L'échantillonnage à la fréquence de Nyquist du champ scalaire ne suffit pas à assurer un échantillonnage correct de l'intégrale de rendu volumique dont la fréquence de Nyquist dépend également des fréquences présentes dans la fonction de transfert [Schulze J.P. *et al.*, 2003, Kraus M., 2003]. En effet, l'approximation réalisée suppose également une émission $c(s)$ et une absorption $\tau(s)$ constantes, lesquelles dépendent de l'étape non linéaire de classification. Comme nous l'avons dit précédemment, il est possible d'affiner la discrétisation de l'intégrale pour échantillonner à la fréquence de Nyquist combinée du champ scalaire et de la fonction de transfert. Cette démarche conduit à une réduction significative des performances. Rezk-Salama *et al.* utilisent dans [Rezk-Salama C. *et al.*, 2000] l'accélération par le matériel graphique pour générer des plans intermédiaires à la volée sur le processeur de fragment dans les techniques de rendu volumique à base de textures 2D. Leur approche réduit les artefacts mais reste marginale. En effet, le nombre de plans nécessaires pour échantillonner correctement l'intégrale dans le cas de pics dans la fonction de transfert est infini. Cet échec de la méthode classique a conduit au développement d'une technique connue sous le nom de *rendu volumique préintégré* [Röttger S. *et al.*, 2000] qui repose sur une étape de classification modifiée tenant compte des variations d'émission et d'opacité liées à une approximation linéaire du champ scalaire dans le segment. Cette technique est à la base de la méthode de rendu des cubes sismiques que nous proposons.

Classification préintégrée

L'origine des idées qui prédominent dans le rendu volumique préintégré est à rechercher dans les modèles optiques développés au début des années 1990 pour la visualisation de grilles non structurées par des techniques de projection [Max N. *et al.*, 1990]. Ces travaux précurseurs ont été successivement repris et améliorés pour affiner le calcul de l'intégrale de rendu volumique [Stein C. *et al.*, 1994, Max N., 1995, Williams P. *et al.*, 1998] et aboutir à la version actuelle du rendu volumique préintégré [Röttger S. *et al.*, 2000].

La principale évolution par rapport au rendu volumique classique est la disparition de l'approximation "champ constant par morceaux" de l'étape de classification au profit d'une véritable intégration de l'émission et de l'opacité de chaque segment basée sur une variation linéaire du champ dans ce dernier. Nous allons tout d'abord nous intéresser à l'émission d'un segment C_i qui s'exprime comme suit :

$$\left| \begin{array}{l} C_i \ = \ \displaystyle\int_{i\Delta}^{(i+1)\Delta} E_i(t)\ \cdot\ e^{-A_i(t)}\ dt \\ \\ \text{avec :} \left\{ \begin{array}{rcl} E_i(t) & = & c\big(s(t)\big) \\ \\ A_i(t) & = & \displaystyle\int_{i\Delta}^{t} \tau\big(s(t')\big)\ dt' \end{array} \right. \end{array} \right. \qquad (3.9)$$

où, comme dans l'Équation (2.4), le terme $E_i(t)$ simule l'émission ponctuelle de rayonnement en chaque particule le long du segment et le terme $A_i(t)$ son atténuation par absorption liée aux particules rencontrées entre le point d'émission et le point d'entrée. Ce dernier, qui tient donc compte de l'atténuation dans le segment, est généralement appelé terme d'*auto-atténuation* (*"self attenuation"* en anglais). En faisant l'approximation que le champ scalaire varie linéairement dans chaque segment avec s_f et s_b respectivement les valeurs en entrée et en sortie du segment (Figure 3.8-B), et en appliquant le changement de variable suivant :

$$t = (i + \omega)\Delta \quad \Leftrightarrow \quad dt = \Delta \; d\omega$$

l'Équation (3.9) devient :

$$\left| \begin{array}{l} C_i \;=\; \displaystyle\int_0^1 E_i(\omega) \;\cdot\; e^{-A_i(\omega)} \;\Delta \; d\omega \\ \\ \text{avec :} \left\{ \begin{array}{rcl} E_i(\omega) & = & c\Big((1-\omega)s_f + \omega s_b\Big) \\ \\ A_i(\omega) & = & \displaystyle\int_0^{\omega} \tau\Big((1-\omega')s_f + \omega' s_b\Big) \;\Delta \; d\omega' \end{array} \right. \end{array} \right. \tag{3.10}$$

où l'émission d'un segment C_i ne dépend que des valeurs du champ scalaire en entrée et en sortie, s_f et s_b, et de sa longueur Δ. De la même manière, l'opacité d'un segment α_i s'exprime :

$$\left| \begin{array}{l} \alpha_i \;=\; 1 - e^{-A_i} \\ \\ \text{avec : } A_i \;=\; \displaystyle\int_{i\Delta}^{(i+1)\Delta} \tau\big(s(t)\big) \; dt \end{array} \right. \tag{3.11}$$

Après hypothèse de variation linéaire du champ scalaire et changement de variable, elle devient :

$$\left| \begin{array}{l} \alpha_i \;=\; 1 - e^{-A_i} \\ \\ \text{avec : } A_i \;=\; \displaystyle\int_0^1 \tau\Big((1-\omega)s_f + \omega s_b\Big) \;\Delta \; d\omega \end{array} \right. \tag{3.12}$$

où l'opacité d'un segment α_i ne dépend également que de s_f, s_b et Δ.

En considérant que la longueur de segment Δ reste constante, hypothèse acceptable dans toutes les méthode de rendu volumique de grilles structurées présentées aux Sections 2.2.2 et 2.3.2, C_i et α_i ne dépendent plus que des valeurs en entrée et en sortie du segment. Il est ainsi possible de préintégrer la fonction de transfert pour tous les couples $\{s_f, s_b\}$ possibles et de stocker le résultat dans une table 2D dite *table de lecture préintégrée* (*"pre-integrated lookup table"* en anglais) qui remplace la fonction de transfert 1D à l'étape de classification. Celle-ci est désormais dite *classification préintégrée*, par opposition aux techniques classiques de pré- et post-classification (Section 2.2.2).

Applications

Le principe de classification préintégrée est un principe générique qui s'applique à toutes les techniques de rendu volumique. En effet, tant les méthodes de discrétisation implicite que

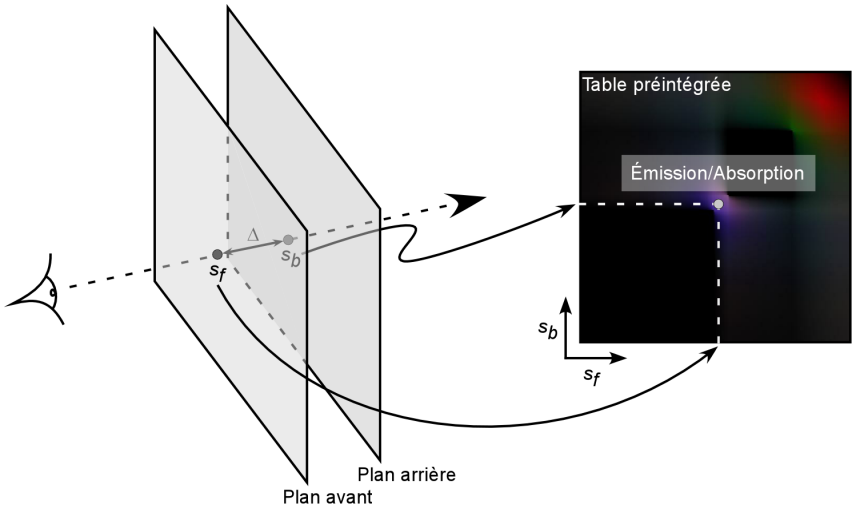

FIG. 3.9 – Le rendu volumique préintégré par méthode implicite à base de textures 3D remplace les plans d'épaisseur nulle de l'approche classique par de véritables tranches épaisses de volume. Les valeurs s_f et s_b échantillonnées sur les plans avant et arrière servent d'entrées dans une table préintégrée d'émissions et d'opacités de segments.

les méthodes de discrétisation explicite, accélérées par le matériel graphique ou non, reposent sur une étape de classification pour laquelle la distinction pré-/post-classification s'applique. Ainsi, cette étape peut légitimement être modifiée pour tenir compte d'une fonction de transfert bidimensionnelle correspondant à la version préintégrée de la fonction de transfert initiale.

La littérature présente donc des exemples de rendu volumique préintégré pour la plupart des techniques présentées au chapitre précédent. Ainsi, le lancer de rayons sans accélération matérielle possède sa version préintégrée [KNITTEL G., 2002], de même que le lancer de rayons avec accélération matérielle [RÖTTGER S. *et al.*, 2003]. Les méthodes de discrétisation implicite ne dérogent pas à la règle, des versions préintégrées ayant été proposées tant pour le "shear-warp" [SCHULZE J.P. *et al.*, 2003] que pour les techniques de projection [WEILER M. *et al.*, 2003]. Des implantations du rendu volumique préintégré existent également pour le matériel graphique spécialisé [MEISSNER M. *et al.*, 2002b]. Une étude exhaustive des fondements de la méthode et de ses retombées en visualisation volumique a été proposée dans [KRAUS M. et ERTL T., 2005].

Dans le cadre de notre travail, il est cependant important de citer les travaux référence de ENGEL *et al.* pour l'application du rendu volumique préintégré au cas des techniques de discrétisation implicite à base de textures 3D [ENGEL K. *et al.*, 2001]. Conceptuellement, la méthode consiste à remplacer les plans d'épaisseur nulle de la technique classique par de véritables tranches épaisses de volume. L'expression *"slab rendering"* apparaît dans la littérature où le terme "slab" (tranche épaisse) s'oppose au terme "slice" (tranche fine). Nous détaillerons cette méthode en Section 3.3.1. En pratique, un programme de fragment échantillonne les valeurs s_f et s_b dans la texture 3D, lesquelles servent de coordonnées pour accéder à une texture 2D qui stocke la table préintégrée. Cette lecture dans la texture 2D est dite *dépendante* (*"dependant texture lookup"* en anglais) du fait que ses entrées dépendent d'une lecture précédente. Le principe de base est illustré sur la Figure 3.9. Nous aurons l'occasion de revenir plus en détails sur l'implantation de cette technique en Section 3.3.1 dans la mesure où elle est à la base de notre moteur de rendu volumique. Les résultats comparés avec l'approche classique à base de post-classification apparaissent sur la Figure 3.10 pour des données de scanner d'un bonsaï et

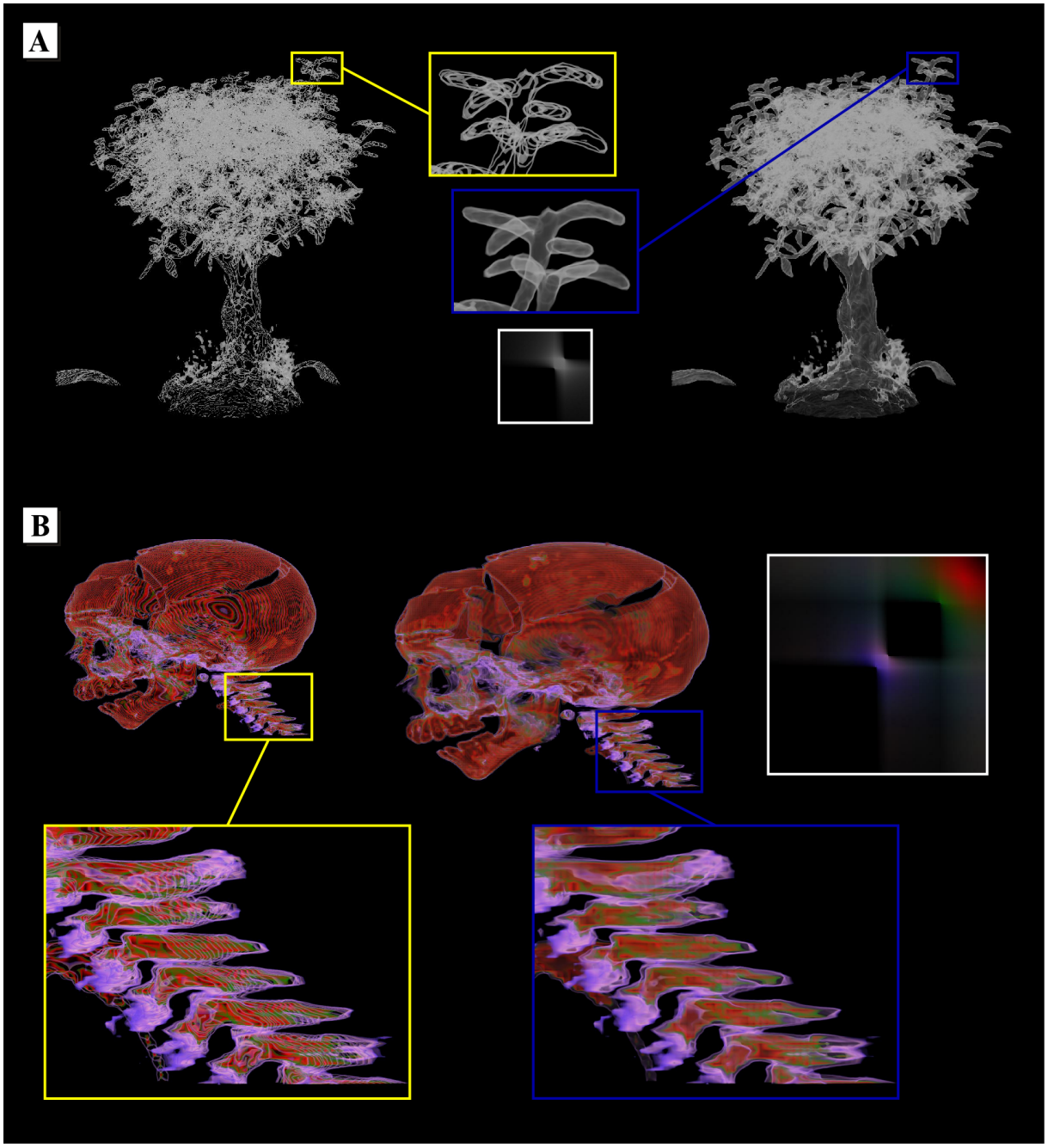

FIG. 3.10 – Comparaison entre post-classification et classification préintégrée pour le rendu volumique par méthode implicite à base de textures 3D sur des données de scanner d'un bonsaï (A) et d'un crâne humain (B). Les images obtenues par la méthode classique de post-classification sont représentées à gauche, tandis que celles de droite ont été obtenues par classification préintégrée. Les rectangles jaunes et bleus montrent respectivement des agrandissements du rendu à base de post-classification et de classification préintégrée. Les tables préintégrées apparaissent dans les carrés blancs.

d'un crâne humain. Il est intéressant de voir comme les deux niveaux de couleur du crâne (bleu et rouge) apparaissent distinctement et en continu sur l'image agrandie des vertèbres visualisée avec préintégration.

Préintégration de la fonction de transfert

Bien qu'il n'intervienne qu'en prétraitement dès lors que l'utilisateur modifie la fonction de transfert, le calcul des intégrales sur les segments $[s_f, s_b]$ lors du remplissage de la table constitue la principale limitation du rendu volumique préintégré. Numériquement, ce calcul repose sur les mêmes simplifications qui conduisent à l'expression discrète de l'intégrale de rendu volumique de l'Équation (2.13), ce qui donne l'expression suivante de l'émission d'un segment :

$$C_i \approx \sum_{s=s_f}^{s_b} C_s \prod_{s'=s_f}^{s-1} (1 - \alpha_{s'}) \tag{3.13}$$

où l'intégrale de l'Équation (3.10) a été décomposée en $|s_f - s_b| + 1$ sous-segments pour échantillonner correctement les fréquences de la fonction de transfert. C_s est l'émission du sous-segment s et $\alpha_{s'}$ l'opacité du sous-segment s'. De manière similaire, l'expression de l'opacité d'un segment fournie à l'Équation (3.12) se décompose comme suit :

$$\alpha_i \approx 1 - \prod_{s=s_f}^{s_b} (1 - \alpha_s) \tag{3.14}$$

Comme pour l'intégrale de rendu volumique discrète globale de l'Équation (2.13), les versions discrètes de l'intégrale dans un segment, tant pour l'émission que pour l'opacité, peuvent être évaluées en accumulant récursivement la contribution des sous-segments. Pour ce faire, il est possible de procéder d'arrière en avant en utilisant l'opérateur de composition "over" de l'Équation (2.14) qui dans ce contexte s'écrit :

$$\left| \begin{array}{rcl} C'_s & = & C_s + (1 - \alpha_s)\, C'_{s+1} \\ \alpha'_s & = & \alpha_s + (1 - \alpha_s)\, \alpha'_{s+1} \end{array} \right. \tag{3.15}$$

où C'_s et α'_s sont respectivement l'émission et l'opacité accumulées entre les sous-segments s_b et s (l'hypothèse $s_f < s_b$ est faite ici). Dans ce cas, l'opacité doit effectivement être accumulée dans la mesure où les valeurs d'émission et d'opacité ainsi calculées serviront ultérieurement lors du calcul de l'intégrale globale de l'Équation (2.13) par accumulation incrémentale de la contribution des segments à l'aide de l'un des opérateurs (2.14) ou (2.15). La composition des sous-segments est également possible de l'avant vers l'arrière, auquel cas elle a recours à l'opérateur "under" de l'Équation (2.15) qui s'écrit alors :

$$\left| \begin{array}{rcl} C''_s & = & C''_{s-1} + (1 - \alpha''_{s-1})\, C_s \\ \alpha''_s & = & \alpha''_{s-1} + (1 - \alpha''_{s-1})\, \alpha_s \end{array} \right. \tag{3.16}$$

où C''_s et α''_s sont respectivement l'émission et l'opacité accumulées entre les segments s_f et s (l'hypothèse $s_f < s_b$ restant valable).

	Méthode brute avec auto-atténuation	Approximation sans auto-atténuation	Sous-intervalles incrémentaux
Athlon 650 MHz	30 s	0.3 s	
Athlon 2.2 GHz	1.57 s		0.05 s
Pentium M 1.69 GHz	0.65 s	0.01 s	0.03 s

TAB. 3.1 – Temps de calcul comparés de la table préintégrée par la méthode brute avec auto-atténuation, l'approximation sans auto-atténuation et l'algorithme des *sous-intervalles incrémentaux* dans le cas $n = 256$. Nos propres implantations correspondent aux temps donnés sur le Pentium M 1.69 GHz tandis que les temps sur les Athlon proviennent de la littérature (Athlon 650 MHz [ENGEL K. *et al.*, 2001] et Athlon 2.2 GHz [LUM E.B. *et al.*, 2004]). Nous obtenons un facteur d'accélération de 22 par la technique des sous-intervalles incrémentaux par rapport à la méthode brute, tout en tenant compte de l'auto-atténuation dans les segments.

À ce stade, il est important de préciser que l'opacité retournée par la fonction de transfert est une grandeur définie par unité de longueur, calibrée par exemple sur la taille d'un voxel. Ainsi, dans le cas du rendu volumique classique avec post-classification, un taux d'échantillonnage constant permet d'utiliser les valeurs d'opacité telles qu'elles sont lues dans la fonction de transfert. En revanche, les calculs de préintégration des segments $[s_f, s_b]$ manipulent des sous-segments de longueur variable qui dépend à la fois de la longueur de segment Δ et du nombre d'échantillons entre s_f et s_b dans la fonction de transfert. Pour obtenir des résultats cohérents dans ce cas, il est donc nécessaire d'appliquer une correction à la valeur d'opacité $\tau(s)$ que retourne la fonction de transfert pour obtenir l'opacité d'un sous-segment α_s. SCHULZE *et al.* reprennent dans [SCHULZE J.P. *et al.*, 2003] la formule de α_i donnée à l'Équation (3.12) en fonction de Δ pour en déduire la valeur de α_s dans un sous-segment de longueur Δ' comme suit :

$$\left| \begin{array}{l} \alpha_s \;=\; 1 - e^{-A_i(\Delta')} \;=\; 1 - \Big(e^{-A_i(\Delta)}\Big)^{\frac{\Delta'}{\Delta}} \;=\; 1 - \Big(1 - \alpha_i\Big)^{\frac{\Delta'}{\Delta}} \\ \\ \text{avec : } A_i(\Delta') \;=\; \displaystyle\int_0^1 \tau\Big((1-\omega)s_f + \omega s_b\Big)\; \Delta'\; d\omega \end{array} \right. \tag{3.17}$$

où Δ' est extrait de l'intégrale pour élever l'exponentielle à la puissance $\frac{\Delta'}{\Delta}$. Ceci permet de retrouver la formule proposée par LUM *et al.* dans [LUM E.B. *et al.*, 2004] qui exprime α_s en fonction de $\tau(s)$:

$$\alpha_s \;=\; 1 - \Big(1 - \tau(s)\Big)^{\frac{d}{|s_f - s_b| + 1}} \tag{3.18}$$

où d est la longueur unitaire des segments de préintégration et où apparaissent légitimement les valeurs en entrée et en sortie de segment s_f et s_b.

Pour n entrées dans la fonction de transfert, le temps de calcul des intégrales discrètes des Équations (3.13) et (3.14) à l'aide de l'un des opérateurs (3.15) ou (3.16) est en $O(n)$. D'autre part, le dimensionnement de la table préintégrée dépend généralement de celui de la fonction de transfert. Ainsi, le temps de remplissage d'une table de taille $n \times n$ par la méthode brute est en $O(n^3)$. ENGEL *et al.* reportent un temps de 30 secondes pour $n = 256$ (cas courant d'un champ scalaire codé sur 8 bits) sur un Athlon 650 MHz [ENGEL K. *et al.*, 2001]. Quelques années plus tard, LUM *et al.* reportent 1.57 secondes pour la même taille de table sur un Athlon FX-51 2.2 GHz [LUM E.B. *et al.*, 2004]. Quant à notre implantation, elle génère la table en 0.65 secondes sur un Pentium M 1.69 GHz. Ces temps sont reportés dans la Table 3.1. Dans tous les cas

et malgré la progression observée, ce temps est encore aujourd'hui trop lent pour assurer une édition confortable en temps réel de la fonction de transfert.

Pour accélérer cette étape de préintégration, Engel *et al.* proposent dans [Engel K. *et al.*, 2001] de négliger l'auto-atténuation dans les segments. Ceci leur permet d'introduire les fonctions intégrales suivantes :

$$\left| \begin{array}{rcl} K(s) & = & \displaystyle\int_0^s c(s')\ ds' \\ \\ T(s) & = & \displaystyle\int_0^s \tau(s')\ ds' \end{array} \right. \tag{3.19}$$

qui sont calculées une seule fois. La fonction $K(s)$ permet de remplacer l'expression de l'émission d'un segment C_i de l'Équation (3.10) par l'approximation suivante :

$$C_i \ \approx \ \frac{\Delta}{s_b - s_f}\Big(K(s_b) - K(s_f)\Big) \tag{3.20}$$

où le terme exponentiel d'auto-atténuation a disparu. De même, l'opacité d'un segment α_i dont l'expression est donnée à l'Équation (3.12) devient :

$$\left| \begin{array}{l} \alpha_i \ = \ 1 - e^{-A_i} \\ \\ \text{avec : } A_i \ = \ \displaystyle\frac{\Delta}{s_b - s_f}\Big(T(s_b) - T(s_f)\Big) \end{array} \right. \tag{3.21}$$

À chaque modification de la fonction de transfert par l'utilisateur, le temps de calcul d'une table préintégrée de taille $n \times n$ à l'aide de ces équations simplifiées est en $O(n^2)$, mesuré à 0.3 secondes par Engel *et al.* sur leur Athlon 650 MHz dans le cas $n = 256$ [Engel K. *et al.*, 2001]. Pour la même taille de table, notre implantation donne un temps de 0.01 secondes sur un Pentium M 1.69 GHz (Table 3.1). Bien que satisfaisante du point de vue des performances qui en résultent, cette simplification conduit dans certains cas à des résultats incorrects, notamment lorsque l'utilisateur place des pics d'opacité maximale rapprochés dans la fonction de transfert. Or, ce cas de figure est fréquent dans les fonctions de transfert types que nous avons décrites pour l'interprétation sismique (Section 3.1.2).

Un premier pas vers le préintégration de la fonction de transfert en temps réel avec prise en compte de l'auto-atténuation dans les segments est basé sur l'utilisation des opérations de blending accélérées par le matériel graphique standard [Röttger S. et Ertl T., 2002]. Dans ce travail, la boucle d'intégration des Équations (3.13) et (3.14) à l'aide de l'opérateur (3.15) est effectuée sur le GPU, le résultat étant directement écrit en mémoire de texture. Cependant, le calcul de la table dans son ensemble, bien qu'accéléré par l'utilisation du processeur graphique, reste en $O(n^3)$.

Nous utilisons donc une technique plus récente baptisée algorithme des *sous-intervalles incrémentaux* [Lum E.B. *et al.*, 2004] qui remplit la table préintégrée en $O(n^2)$ tout en tenant compte de l'auto-atténuation. Les auteurs partent du constat que la correction d'opacité de l'Équation (3.18) à appliquer lors de l'étape de préintégration est une correction non linéaire qui interdit donc l'utilisation du résultat de l'intégration entre s_f et s_b pour le calcul incrémental de l'intégration entre s_f et $s_b + 1$. En effet, de cette non linéarité découle l'obligation d'effectuer à nouveau l'ensemble des opérations de composition de sous-segments avec les nouvelles valeurs corrigées d'opacité. En revanche, dès lors que le nombre de sous-segments à composer $|s_f - s_b| + 1$

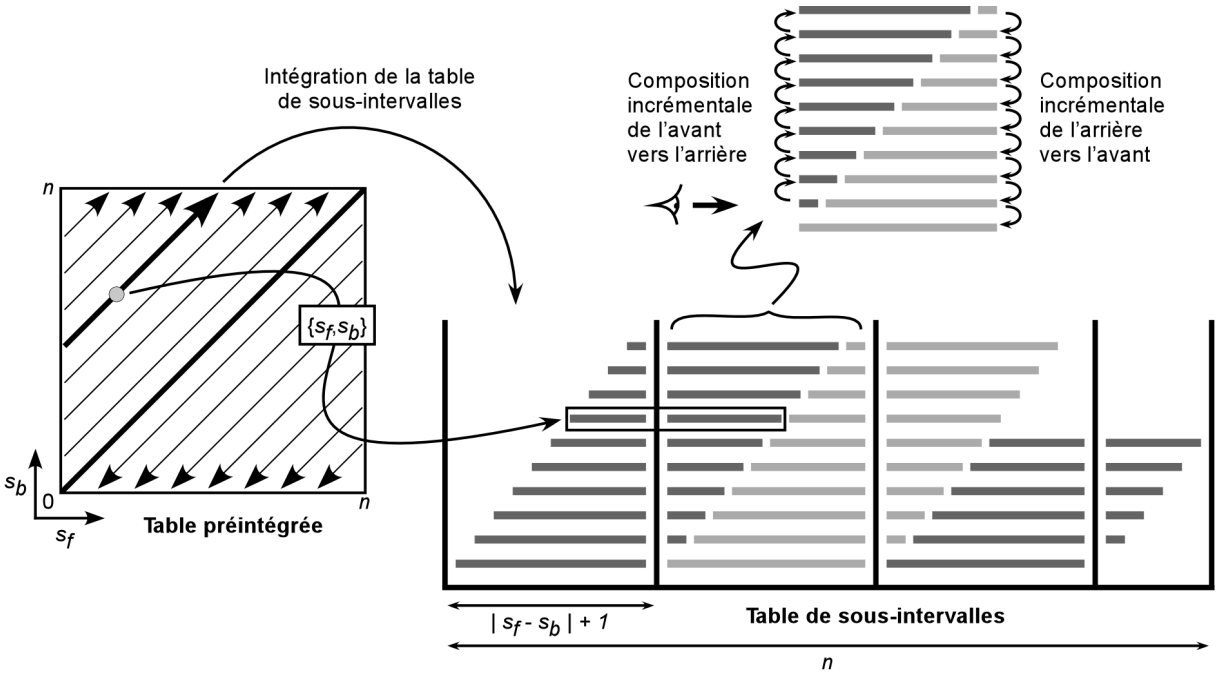

FIG. 3.11 – Préintégration de la fonction de transfert à l'aide de l'algorithme des *sous-intervalles incrémentaux*. Le remplissage de la table préintégrée est effectué diagonale par diagonale, formant ainsi des groupes de segments $[s_f, s_b]$ dont le nombre de sous-segments est identique. Pour chaque diagonale, une table de sous-intervalles est intégrée incrémentalement en un temps $O(n)$ où n est la taille de la fonction de transfert. Les résultats intermédiaires sont stockés et, pour tout couple $\{s_f, s_b\}$ dans la diagonale, les deux sous-intervalles correspondants sont composés. Pour une table de taille $n \times n$, le calcul des $2n-1$ diagonales se fait donc en un temps $O(n^2)$ (modifié d'après [LUM E.B. *et al.*, 2004]).

reste constant, la correction à appliquer est la même. L'idée est donc de calculer les intégrales des Équations (3.13) et (3.14) par groupes de segments $[s_f, s_b]$ dont le nombre de sous-segments est identique, ce qui revient à remplir la table préintégrée diagonale par diagonale. De cette manière, comme le montre la Figure 3.11 pour une diagonale particulière, il est possible d'intégrer incrémentalement une table de sous-intervalles en un temps $O(n)$. En conservant les résultats de l'intégration des sous-intervalles intermédiaires, cette table permet de calculer l'intégration pour chaque couple $\{s_f, s_b\}$ de la diagonale par la simple composition de deux sous-intervalles. Les $2n-1$ diagonales d'une table de taille $n \times n$ se calculent donc en un temps $O(n^2)$. Dans le cas $n = 256$, LUM *et al.* reportent un temps de 0.05 secondes sur un Athlon FX-51 2.2 GHz [LUM E.B. *et al.*, 2004]. Notre implantation de l'algorithme génère la table préintégrée en 0.03 secondes sur un Pentium M 1.69 GHz, ce qui représente un facteur d'accélération de 22 par rapport à la méthode brute en $O(n^3)$ (Table 3.1).

3.2.3 Préintégration et illumination locale

Dans le cas de l'application d'un modèle d'illumination locale, le résultat de l'intégrale de rendu volumique dépend non seulement des valeurs du champ scalaire rencontrées mais également des directions du gradient. Tel qu'il a été présenté jusqu'ici, le rendu volumique préintégré est basé sur la seule prise en compte des valeurs en entrée et en sortie de segment pour en déduire le résultat de l'intégration. Une préintégration des effets d'illumination reposerait sur la prise

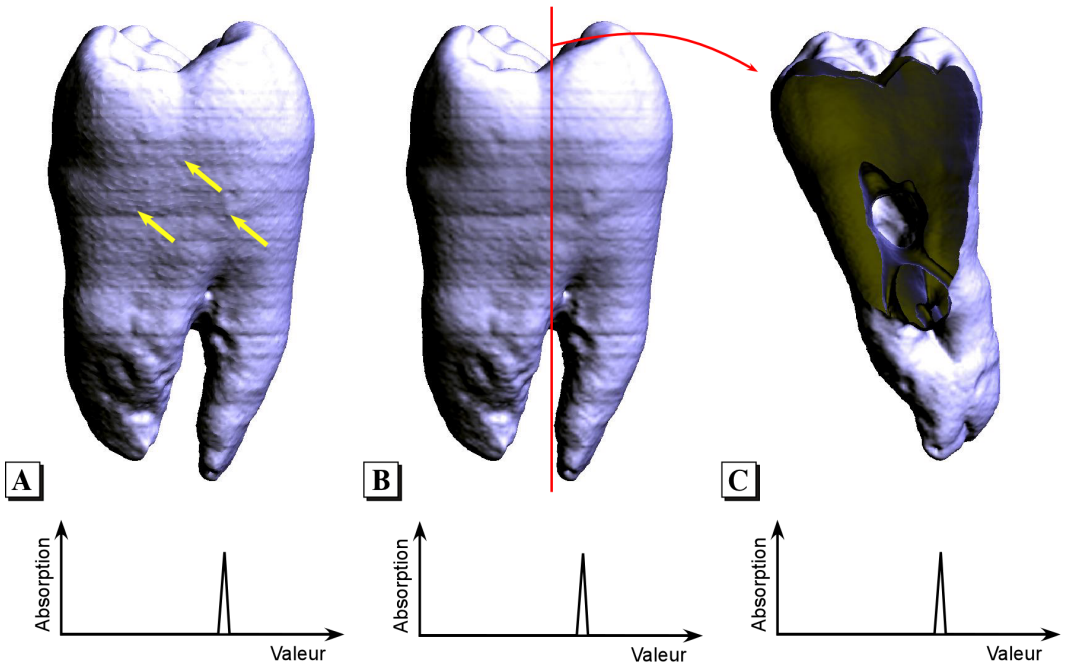

FIG. 3.12 – Rendu volumique par méthode implicite à base de textures 3D sur des données de scanner d'une dent avec classification préintégrée et illumination locale. Calcul des effets d'illumination par estimation ponctuelle du gradient (A) ou *illumination préintégrée interpolée* (B,C). Les hautes fréquences introduites par un pic d'opacité dans la fonction de transfert sont correctement reproduites par la classification préintégrée (à comparer aux résultats de post-classification de la Figure 3.5-A). La coupe transversale confirme le fait que le rendu porte sur la seule enveloppe superficielle de la dent. D'autre part, l'approximation par illumination préintégrée interpolée supprime les artefacts liés à l'approche naïve par estimation ponctuelle du gradient (flèches jaunes).

en compte supplémentaire des directions du gradient en entrée et en sortie, ce qui nécessite le calcul d'une table de dimension huit à la place de la table 2D actuellement utilisée. Au vu de la discussion du paragraphe précédent, il est aisé d'imaginer les conséquences catastrophiques sur les temps de calcul.

Estimation ponctuelle et sur-échantillonnage

Pour simplifier la situation, ENGEL *et al.* se contentent d'une estimation ponctuelle du gradient dans la tranche (moyenne des gradients en entrée et en sortie) à partir de laquelle ils appliquent l'effet de l'illumination sur l'émission préintégrée [ENGEL K. *et al.*, 2001]. Cette approche repose sur la table préintégrée 2D classique et bénéficie donc de la préintégration de l'émission et de l'opacité, tandis que le calcul d'illumination reste ponctuel. Comme le montre la Figure 3.12-A, ceci conduit inévitablement à des artefacts marquant la transition entre deux tranches, bien que l'émission et l'opacité aient été correctement échantillonnées (à comparer aux résultats de post-classification de la Figure 3.5-A). Pour le cas particulier de pics d'opacité dans la fonction de transfert, une approche similaire mais plus évoluée a été proposée [MEISSNER M. *et al.*, 2002a]. Dans ce cas, les auteurs utilisent une interpolation des gradients en entrée et en sortie par un facteur dépendant de coefficients pondérés par les valeurs d'opacité de ces pics. Pour revenir au cas général, RÖTTGER *et al.* proposent dans [RÖTTGER S. *et al.*, 2003]

de sur-échantillonner les calculs d'illumination. Un échantillonnage quatre fois supérieur à la fréquence de Nyquist du champ scalaire s'est avéré suffisant dans leurs expériences. Cependant, cette approche empirique d'une part réduit les performances du fait du sur-échantillonnage, d'autre part ne produit pas un résultat satisfaisant dans tous les cas.

Illumination préintégrée interpolée

Nous utilisons donc la technique plus récente d'*illumination préintégrée interpolée* proposée dans [Lum E.B. *et al.*, 2004]. L'objectif visé est de produire une image cohérente au sens où elle ne doit faire apparaître de changements brusques de couleur à la surface d'une structure d'intérêt (comme l'enveloppe superficielle de la dent de la Figure 3.12) ni lors du passage de cette dernière d'une tranche d'échantillonnage à une autre, ni lors d'un changement de point de vue. Cet objectif est atteint par interpolation des effets d'illumination entre l'avant et l'arrière d'un segment. Bien que ne fournissant qu'une approximation de la préintégration réelle des effets d'illumination (non envisageable dans la pratique), cette interpolation a le mérite de produire des images cohérentes comme en témoignent les Figures 3.12-B et 3.12-C. Toute la difficulté de l'approche réside dans la façon de combiner l'interpolation de l'illumination avec le principe de préintégration de l'émission et de l'opacité basée sur la table de lecture préintégrée.

Nous allons tout d'abord décrire la méthode dans le cas simplifié d'un modèle d'illumination purement diffus où les effets spéculaires sont négligés. D'autre part, l'illumination n'affectant pas l'opacité, nous ne tiendrons compte que de l'émission. La démarche de Lum *et al.* consiste à effectuer deux calculs d'illumination par segment, l'un pondéré vers l'avant du segment, l'autre pondéré vers l'arrière, puis de faire la somme des résultats pour obtenir l'illumination finale. Pour ce faire, ils remplacent la table préintégrée 2D classique par deux tables, l'une dans laquelle la contribution des sous-segments est pondérée vers l'avant du segment, l'autre vers l'arrière, et de telle sorte que la somme segment par segment des deux tables donne la table classique. Ainsi, pour un segment $[s_f, s_b]$ la table classique stocke l'approximation de l'émission préintégrée C_i donnée à l'Équation (3.13), approximation que nous appellerons $C(s_f, s_b)$ et dont nous simplifierons l'écriture comme suit :

$$\left| \begin{array}{l} C(s_f, s_b) \;=\; \displaystyle\sum_{s=s_f}^{s_b} C_s \; A_s \\ \\ \text{avec : } A_s \;=\; \displaystyle\prod_{s'=s_f}^{s-1} (1 - \alpha_{s'}) \end{array} \right. \tag{3.22}$$

où C_s est l'émission du sous-segment s et $\alpha_{s'}$ l'opacité du sous-segment s'. Le terme A_s correspond à l'auto-atténuation le long du segment du rayonnement émis par le sous-segment s. De la même manière, pour un segment $[s_f, s_b]$ les tables pondérées stockent une approximation de l'intégration pondérée de l'émission vers l'avant et vers l'arrière que nous noterons respectivement $C^f(s_f, s_b)$ et $C^b(s_f, s_b)$, et qui s'expriment comme suit :

$$\left| \begin{array}{l} C^f(s_f, s_b) \;=\; \displaystyle\sum_{s=s_f}^{s_b} \frac{s - s_b}{s_f - s_b} \, C_s \; A_s \\ \\ C^b(s_f, s_b) \;=\; \displaystyle\sum_{s=s_f}^{s_b} \frac{s - s_f}{s_b - s_f} \, C_s \; A_s \end{array} \right. \tag{3.23}$$

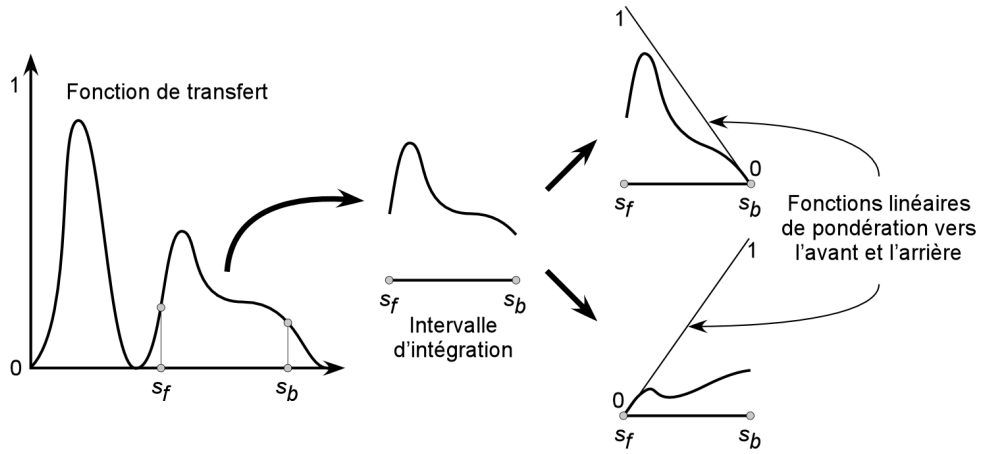

FIG. 3.13 – Principe de l'intégration pondérée dans la technique d'*illumination préintégrée interpolée*. La contribution de l'émission des sous-segment dans la table classique est multipliée par une fonction rampe orientée selon le cas vers l'avant ou l'arrière du segment (modifié d'après [LUM E.B. *et al.*, 2004]).

ce qui permet bien de retrouver l'égalité :

$$C \;=\; C^f + C^b$$

Comme l'illustre la Figure 3.13 pour un segment $[s_f, s_b]$ particulier, la contribution de l'émission des sous-segments dans la table classique est multipliée par une fonction rampe orientée selon le cas vers l'avant ou l'arrière du segment. Par exemple, celle du sous-segment s_f est maximale dans la table pondérée vers l'avant (multipliée par 1) et minimale dans la table pondérée vers l'arrière ou elle s'annule, et inversement pour le sous-segment s_b. À partir de ces deux tables, deux calculs d'illumination sont effectués par segment, l'un tient compte du gradient en entrée et utilise la table pondérée vers l'avant, inversement l'autre tient compte du gradient en sortie et utilise la table pondérée vers l'arrière. Le résultat est un calcul d'illumination progressivement interpolé qui produit des images cohérentes, comme en témoigne la Figure 3.12-B où les artefacts visibles sur la Figure 3.12-A ont disparu.

Pour calculer ces tables pondérées de manière efficace, LUM *et al.* introduisent une table globalement pondérée dans laquelle l'intégration de l'émission d'un segment sera notée $C^G(s_f, s_b)$ et s'exprime comme suit :

$$C^G(s_f, s_b) \;=\; \sum_{s=s_f}^{s_b} s\; C_s\; A_s \tag{3.24}$$

Cette table permet de modifier les expressions de l'intégration pondérée vers l'avant et vers l'arrière de l'Équation (3.23) comme suit :

$$\left|\begin{array}{rcl} C^f(s_f, s_b) & = & \dfrac{C^G(s_f, s_b) - s_b\; C(s_f, s_b)}{s_f - s_b} \\[2ex] C^b(s_f, s_b) & = & \dfrac{C^G(s_f, s_b) - s_f\; C(s_f, s_b)}{s_b - s_f} \end{array}\right. \tag{3.25}$$

L'intégration des deux tables pondérées respectivement vers l'avant et vers l'arrière ne nécessite donc que le calcul de la table préintégrée classique ainsi que d'une table globalement pondérée,

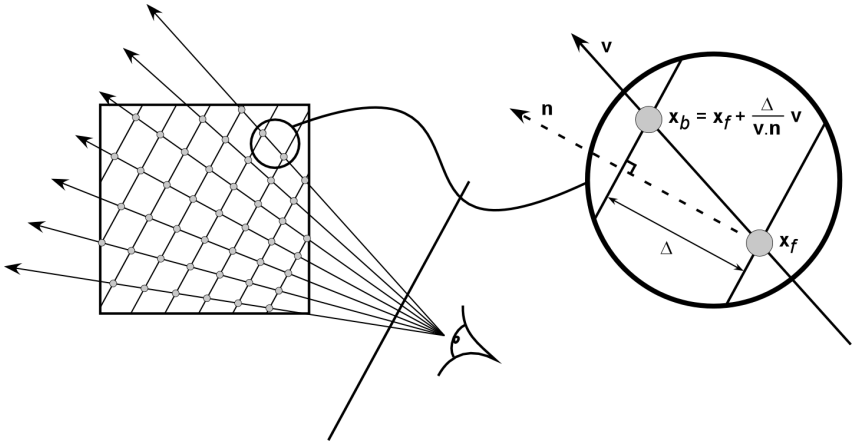

FIG. 3.14 – Principe du calcul des coordonnées du point sur le plan arrière $\mathbf{x}_b$ dans le programme de sommet pour le rendu volumique préintégré. Ce calcul repose sur la direction d'observation $\mathbf{v}$, la normale aux plans d'échantillonnage $\mathbf{n}$, la position du point sur le plan avant $\mathbf{x}_f$ et la distance entre deux plans qui n'est autre que la longueur Δ des segments de préintégration.

obtenue de la même manière que la table classique en remplaçant la fonction de transfert initiale par une fonction de transfert où les valeurs d'émission $c(s)$ ont été multipliées par s. En utilisant la technique des sous-intervalles incrémentaux, il suffit donc de 0.06 secondes sur notre Pentium M 1.69 GHz. Dans le cas où les effets spéculaires sont pris en compte, il est nécessaire de rajouter deux tables pondérées, l'une vers l'avant l'autre vers l'arrière, et construites à partir d'une fonction de transfert dont toutes les émissions sont de couleur blanche. Le temps de prétraitement lors d'une modification de la fonction de transfert par l'utilisateur dans le cas d'un calcul d'illumination locale diffuse et spéculaire sur notre Pentium M 1.69 GHz est donc de 0.12 secondes, ce qui permet une manipulation de la fonction de transfert en temps réel.

3.3 Notre implantation du rendu volumique préintégré

Dans cette section, nous allons donner les détails de notre implantation du rendu volumique préintégré avec application d'un modèle d'illumination locale. Celle-ci est destinée à être exécutée sur une station d'interprétation classique disposant d'une carte graphique standard. Elle a été implantée dans le module VolumExplorer du logiciel Gocad par l'ajout d'un nouveau type de sondes nommé *Volume* dont le principe est identique à celui des sondes de type Skin (Section 1.3), mais en remplaçant le rendu de la frontière du sous-volume par du rendu volumique dans le sous-volume. Nous verrons les résultats obtenus avec cette méthode pour le rendu volumique du signal sismique.

3.3.1 Mise en œuvre sur le matériel graphique standard

Notre moteur de rendu volumique est fondé sur une approche par discrétisation implicite à base de textures 3D (Section 2.3.2). Comme nous l'avons vu, l'extension de cette méthode au rendu volumique préintégré a été proposée par ENGEL *et al.* dans [ENGEL K. *et al.*, 2001] avec leur technique de rendu volumique par tranches épaisses, appelée "slab rendering". D'autre part, nous utilisons la méthode d'illumination préintégrée interpolée [LUM E.B. *et al.*, 2004] introduite à la section précédente. Nous allons tout d'abord nous intéresser à la mise en œuvre du principe

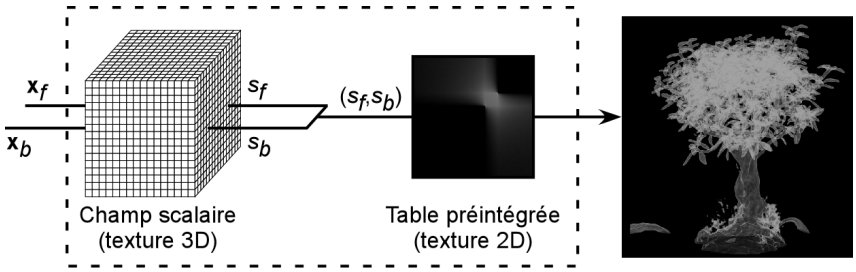

FIG. 3.15 – Principe de lecture dépendante dans la table préintégrée dans le programme de fragment pour le rendu volumique préintégré. Les paramètres en entrée sont les coordonnées de texture 3D des points sur les plans avant et arrière, respectivement $\mathbf{x}_f$ et $\mathbf{x}_b$. La lecture des émission et opacité préintégrées dans la texture 2D qui stocke la table préintégrée dépend du résultat de la lecture des valeurs du champ scalaire dans la texture 3D.

de classification préintégrée, puis nous verrons comment le faire évoluer pour introduire les calculs d'illumination.

Rendu volumique par tranches épaisses

Dans la technique classique de rendu volumique par méthode implicite à base de textures 3D, les données volumiques sont stockées dans la mémoire de texture et des plans d'échantillonnage sont dessinés par la carte graphique (Section 2.3.2). À chaque sommet des polygones définissant ces plans sont affectées des coordonnées de texture. Après le passage par le processeur de sommet, les polygones sont décomposés en fragments à l'étape de rasterisation (avec interpolation linéaire des coordonnées de texture) et chaque fragment est traité par le processeur de fragment qui applique l'émission et l'opacité correspondant à la valeur scalaire lue dans la texture. Les fragments sont composés au niveau des opérations de fragment qui appliquent l'"alpha blending" et écrivent le résultat dans la mémoire écran. La contribution de la matière qui se situe entre les plans d'épaisseur nulle ainsi dessinés n'est pas prise en compte. En effet, l'échantillonnage au niveau du plan applique l'approximation "champ constant par morceaux" de l'Équation (2.12) avec les conséquences que nous avons pu voir, à savoir des problèmes de sous-échantillonnage de l'intégrale de rendu volumique dans le cas de fonctions de transfert à fréquence de Nyquist élevée.

Conceptuellement, ENGEL *et al.* proposent de remplacer ces plans d'épaisseur nulle par de véritables tranches épaisses de volume en considérant qu'un plan et son suivant définissent une tranche. Ainsi, comme le montre la Figure 3.9, les valeurs en entrée et en sortie de tranche peuvent être prises en compte pour lire l'émission et l'opacité dans une table préintégrée. En pratique, nous avons mis ceci en œuvre en combinant un programme de sommet et un programme de fragment. Les données volumiques sont stockées dans une texture 3D tandis que la table préintégrée calculée par la technique des sous-intervalles incrémentaux (Section 3.2.2) est stockée dans une texture 2D. Ceci nécessite le support d'au moins deux unités de texture (Section 2.3.1) dont disposent largement la plupart des cartes graphiques modernes. Le programme de fragment exécute la lecture de cette table à partir des valeurs scalaires lues dans la texture 3D tandis que le programme de sommets intervient en amont pour calculer les coordonnées de texture du point sur le plan arrière à partir de celles du point sur le plan avant. Celles-ci sont interpolées lors de l'étape de rasterisation. Le programme de sommet n'est pas indispensable ici, l'ensemble des opérations pouvant être exécutées dans le programme de fragment. Cependant le processeur de

fragment est limitant en rendu volumique et il est donc intéressant de déléguer ce qui peut l'être au processeur de sommet. La Figure 3.14 illustre le principe de ce calcul dans le cas général d'une vision en perspective. Les données en entrée du programme de fragment sont les coordonnées des points sur le plan avant et sur le plan arrière qui permettent la lecture des valeurs scalaires en entrée et en sortie de tranche dans la texture 3D. Ces valeurs servent ensuite de coordonnées de texture pour lire l'émission et l'opacité préintégrées dans la texture 2D. Ce type de lecture de texture dépendant des résultats de lectures précédentes est illustré sur la Figure 3.15. De cette manière, nous réduisons au maximum le nombre d'instructions exécutées sur le processeur de fragment. En effet, deux instructions sont nécessaires pour lire les valeurs scalaires en entrée et en sortie de tranche dans la texture 3D, puis une instruction supplémentaire pour lire l'émission et l'opacité dans la texture 2D.

Illumination préintégrée interpolée

L'introduction de l'illumination préintégrée interpolée dans le système mis en place ne bouleverse pas le principe général. En effet, les données sont toujours stockées dans une texture 3D tandis que la texture 2D qui stocke la table préintégrée est remplacée par quatre textures qui stockent les tables diffuses et spéculaires pondérées vers l'avant et vers l'arrière (Section 3.2.3). Ceci porte à cinq le nombre d'unités de texture (Section 2.3.1) nécessaire qui, une fois de plus, n'est pas limitant sur la plupart des cartes graphiques modernes. En plus du calcul du point sur le plan arrière, le programme de sommet est en charge du calcul des vecteurs $\mathbf{l}$ et $\mathbf{h}$ qui interviennent dans le modèle d'illumination de Blinn-Phong (Section 3.2.1). Le programme de fragment subit de plus profondes modifications dans la mesure où il est désormais responsable du calcul du modèle d'illumination. Après estimation du gradient du champ scalaire $\mathbf{g}$ aux points des plans avant et arrière, les Équations (3.5) et (3.7) servent respectivement à calculer les contributions diffuse et spéculaire en chacun de ces points. Dans ces équations, le facteur d'émission $e(\mathbf{x})$ est obtenu par lecture des tables préintégrées. Les tables pondérées vers l'avant sont utilisées pour le point sur le plan avant, et inversement. En dernier lieu, les résultats de l'illumination pondérée vers l'avant et vers l'arrière sont sommés pour obtenir l'émission finale. L'opacité n'étant quant à elle pas affectée par le modèle d'illumination (Section 3.2.1), il suffit qu'elle soit stockée dans une seule des tables pondérées et directement appliquée sans modification.

3.3.2 Résultats pour la visualisation du signal sismique

Afin d'évaluer objectivement l'apport du rendu volumique préintégré avec illumination locale pour la visualisation des données sismiques, nous avons appliqué ce nouveau mode de rendu à la sonde de la Figure 3.4 pour une fonction de transfert identique. Les résultats obtenus avec et sans modèle d'illumination sont ceux de la Figure 3.16. Grace à la classification préintégrée, l'intégrale de rendu volumique est correctement calculée sans même sur-échantillonner le champ scalaire et malgré les hautes fréquences introduites par le pic d'opacité de la fonction de transfert. Il est également important de noter l'apport du modèle d'illumination locale préintégrée interpolée sans lequel la forme des structures dans l'espace n'est pas identifiable. Nous montrons ici le cas extrême de l'introduction d'un pic d'opacité. Il est important de noter que la réduction des fréquences dans la fonction de transfert est une pratique courante pour la visualisation volumique des données sismiques. Cependant, comme nous avons pu le voir en Section 3.1.2, cette démarche d'une part constitue un obstacle à l'adoption des techniques de rendu volumique pour

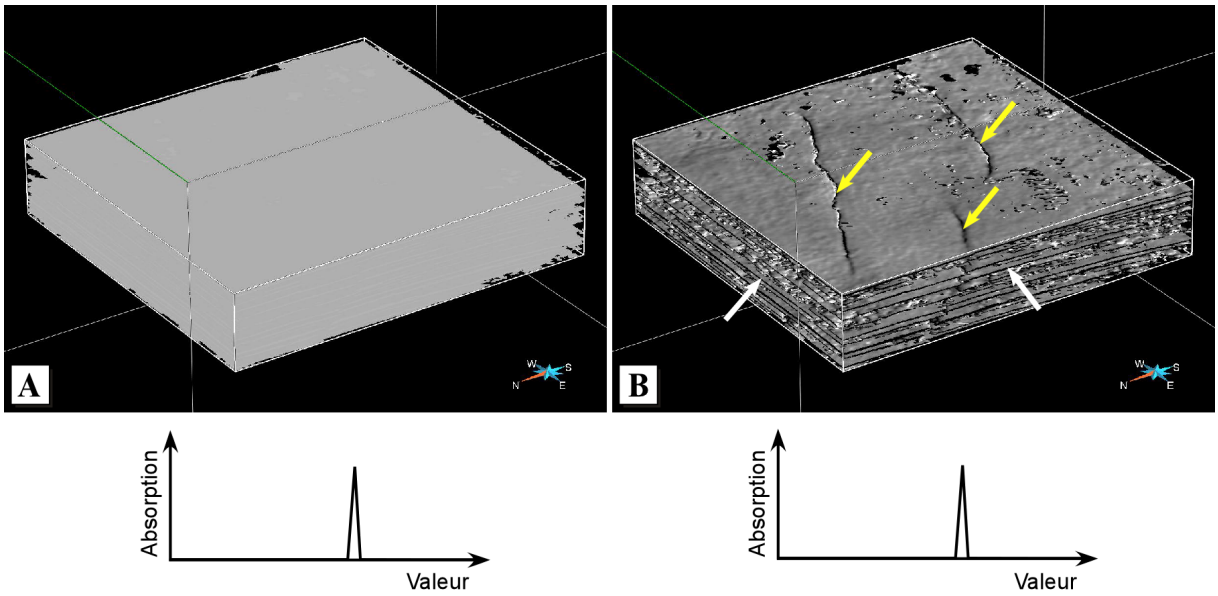

FIG. 3.16 – Rendu volumique par méthode implicite à base de textures 3D avec classification préintégrée sur des données sismiques sans (A) et avec (B) application d'un modèle d'illumination locale. Les courbes d'opacité sont données. Malgré l'introduction de hautes fréquences dans la fonction de transfert (pic d'opacité), l'échantillonnage à la fréquence de Nyquist du champ scalaire (un plan par voxel en moyenne) est suffisant avec la classification préintégrée (à comparer aux résultats de post-classification avec sur-échantillonnage de la Figure 3.4). D'autre part, l'utilisation d'un modèle d'illumination locale préintégrée interpolée permet de mettre clairement en évidence des événements sismiques tels que des horizons (flèches blanches) dont la continuité est affectée par des failles (flèches jaunes).

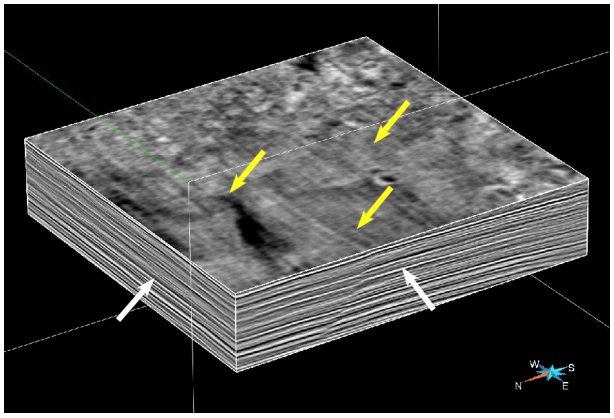

FIG. 3.17 – Visualisation de données sismiques par sonde surfacique. Les flèches blanches et jaunes de la Figure 3.16 qui pointaient respectivement sur des horizons et des failles ont été reportées pour comparaison.

le néophyte, d'autre part en réduit la flexibilité dans l'intervalle d'amplitudes d'intérêt. Il s'agit, à notre avis, des raisons principales de sa faible popularité dans le milieu de l'interprétation sismique. Il nous semble que des techniques telles que le rendu volumique préintégré avec illumination locale, simples d'accès du point de vue de l'utilisateur, puissent constituer une étape significative vers l'adoption du rendu volumique par le plus grand nombre dans le domaine de l'exploration pétrolière.

Pour cela, il est d'autre part important d'évaluer l'apport du rendu volumique par rapport aux techniques classiques de visualisation à base d'outils 2D. La Figure 3.17 montre la même sonde que celle de la Figure 3.16 mais dont seule la frontière est visualisée (type Skin). À la comparaison de ces deux images, l'intérêt d'intégrer le rendu volumique dans la démarche d'interprétation des données sismiques est évident. En effet, la plupart des structures d'intérêt présentent une forme arbitraire dans l'espace qui rend leur mise en évidence par des outils 2D complexe voire impossible dans certains cas. Bien qu'un œil expérimenté parvienne à en déceler les grandes lignes à l'aide des seuls outils habituels, le rendu volumique nous semble être un excellent complément potentiellement capable de réduire le temps d'interprétation.

3.4 Bilan et perspectives

Une étude fine de la structure du signal sismique et de la distribution spatiale des valeurs d'amplitude qui en découle nous a permis de mettre en évidence les principales raisons de l'échec des techniques classiques de rendu volumique dans le contexte de l'interprétation structurale. Bien connues du domaine de la visualisation, ces limitations de la méthode classique sont résolues par des techniques modernes de rendu volumique dont les évolutions du matériel graphique depuis le début des années 2000 rendent l'utilisation possible en pratique et d'un point de vue industriel. La mise en œuvre de ces techniques dans le module VolumeExplorer du logiciel commercial Gocad nous a finalement permis de confirmer leur potentiel à occuper une place de choix dans la démarche d'interprétation des données sismiques. En effet, non seulement elles apportent une réponse aux principales limitations de l'approche classique en rendu volumique, mais elles sont également capables de mettre en évidence de manière triviale la structure tridimensionnelle d'événements sismiques difficilement décelable par l'utilisation des seuls outils 2D d'interprétation.

D'un point de vue technique, nous avons évoqué au cours de ce chapitre l'intérêt que peuvent représenter les fonctions de transfert multidimensionnelles (Section 3.1.2). En effet, la sélection des valeurs d'opacité en fonction de plusieurs sources d'information, comme la valeur d'amplitude sismique et sa dérivée verticale [Gerhardt A. *et al.*, 2001, Gerhardt A. *et al.*, 2002], permet une plus grande flexibilité et surtout une plus grande précision dans la mise en évidence des événements importants. Malheureusement, l'utilisation en pratique de ce type de fonctions de transfert dans le cadre du rendu volumique préintégré reste un problème ouvert en visualisation scientifique. En effet, la dimension du domaine d'intégration devient prohibitive. Cependant, Kniss *et al.* introduisent dans [Kniss J. *et al.*, 2003b] une technique basée sur l'approximation de la courbe d'opacité par une somme de fonctions gaussiennes, donnant ce qu'ils nomment des *fonctions de transfert gausiennes* ou *GTF* (de l'anglais "Gaussian Transfer Functions"). La simplicité de ces fonctions permet de les intégrer de manière analytique et en temps réel sur les processeurs graphiques modernes. Ils sont ainsi capables d'intégrer analytiquement des fonctions de transfert multidimensionnelles sur le processeur graphique. Bien qu'encore limitée dans le cas général de données multi-attributs, la technique peut être simplifiée et spécialisée

pour atteindre le temps réel dans le cas de la combinaison des valeurs d'un champ scalaire et de son gradient. Permettant facilement de représenter une série de pics d'opacité, cette méthode d'approximation par une somme de fonctions gaussiennes est parfaitement adaptée au type de fonctions de transfert que nous manipulons pour l'interprétation structurale du signal sismique. Parallèlement à la combinaison des valeurs d'amplitude sismique et de leur dérivée, nous avons mis en évidence en Section 3.2.1 l'intérêt de remplacer la dérivée verticale du signal par l'attribut de phase instantanée (Section 4.1.1) qui reflète plus fidèlement l'orientation des événements sismiques [Silva P.M. *et al.*, 2003]. Bien que sortant du cas spécialisé de la technique de Kniss *et al.*, il nous semblerait intéressant d'explorer les possibilités d'une telle approche.

Pour conclure sur ce chapitre, nous voudrions souligner l'importance de la perception qui est donnée du rendu volumique à l'utilisateur d'applications d'interprétation sismique. En effet, ce dernier attache finalement moins d'importance aux fondements scientifiques de la technique qu'aux résultats qu'elle va lui fournir. L'un des facteurs limitant son adoption par le plus grand nombre est, comme nous l'avons précédemment souligné (Section 3.1.2), la difficulté d'acquérir les connaissances de base nécessaires à l'édition effective de la fonction de transfert. Bien que de nombreux travaux aient cherché à simplifier la tâche par une approche semi-automatique [Pfister H. *et al.*, 2001, Kindlmann G. et Durkin J., 1998], leur démarche reste empirique et difficilement généralisable. Les travaux présentés dans ce chapitre s'orientent clairement vers une plus grande simplicité d'utilisation, il n'en reste pas moins nécessaire de mettre en œuvre des interfaces d'édition de fonctions de transfert qui soient moins dictées par la technique de rendu utilisée. Le cas des fonctions de transfert multidimensionnelles est significatif. En effet, l'interface d'édition introduite dans [Kniss J. *et al.*, 2002a] par exemple laisse une liberté remarquable au prix d'une période d'apprentissage qui peut paraître dissuasive. L'introduction des fonctions de transfert gaussiennes par ces derniers [Kniss J. *et al.*, 2003b] leur a permis d'abstraire complètement l'interface d'édition de la technique de rendu sous-jacentes. Ainsi, l'utilisateur se contente de sélectionner un certain nombre de structures clés observées sur de simples coupes 2D du volume et de leur associer une couleur. La somme de gaussiennes est automatiquement générée et leur moteur de rendu volumique dispose des informations nécessaires sans que l'utilisateur n'ait eu à aucun moment la lourde tâche d'édition des courbes de la fonction de transfert. Ce dernier point nous semble décisif pour permettre de populariser le rendu volumique dans le domaine de l'interprétation sismique.

Ces considérations liées à l'*interface utilisateur* vont nous permettre d'effectuer la transition vers le chapitre suivant. L'objectif de ce dernier est de décrire des moyens simples et efficaces de combiner plusieurs sources d'information pour optimiser l'interprétation des données sismiques. Pour aller dans le sens de ce qui vient d'être discuté, nous avons par exemple mis au point une interface multimodale générique de sélection et de combinaison de plusieurs volumes d'information. Cette interface constitue un moyen intuitif de combiner les données indépendamment des techniques sous-jacentes de rendu.

Chapitre 4

Visualisation sismique multimodale

Ce chapitre aborde le problème de la multimodalité de l'interprétation sismique, terme faisant référence à la coexistence de plusieurs sources d'information dans le processus d'interprétation. De nature diverse, sismique ou structurale, ce surplus d'information présente un intérêt évident à condition de disposer d'outils de visualisation capables d'en explorer le contenu de manière intelligible et efficace. C'est l'objectif que nous tenterons d'atteindre dans ce chapitre. Nous proposerons tout d'abord une classification des principaux cas de multimodalité en sismique qui seront abordés séparément par la suite. Ainsi, les détails de mise en œuvre d'un système générique de visualisation volumique multimodale seront exposés. Notre système offre de nouveaux modes d'interaction en généralisant la notion de calque des applications de dessin 2D aux données volumiques et permet de combiner différentes sources d'information en temps réel avec un maximum de flexibilité. Par exemple, nous verrons comment l'utilisateur peut facilement et intuitivement inventer de nouveaux modes de visualisation en combinant ces différents calques. Considérant les coupes horizon comme un cas particulier de visualisation multimodale, nous présenterons ensuite un système spécialisé pour la visualisation de coupes horizon en perspective.

4.1 Interprétation sismique multimodale

Comme nous avons pu le voir au cours du Chapitre 1, l'objectif de l'interprétation sismique est d'extraire l'information sur le sous-sol contenue dans le signal sismique. Les différentes formes d'analyse de l'image sismique (stratigraphique, structurale ou pétrophysique) procèdent de manière similaire en examinant le signal à différentes échelles à la recherche de signatures d'événements remarquables. Au cours de ce processus d'interprétation, l'information sismique initiale s'enrichit des observations effectuées. Ainsi, de nouvelles sources d'information sont produites dont la combinaison peut concourir à améliorer le résultat final de l'interprétation et en réduire la durée. Cette combinaison définit le caractère multimodal de l'interprétation sismique. Nous distinguons plusieurs cas de multimodalité selon que l'espace de définition des informations combinées est identique ou différent.

4.1.1 Information multimodale volumique

Le premier cas de multimodalité que nous aborderons concerne l'existence de sources d'information diverses définies sur le même espace volumique que le signal sismique initial. Qu'elle soit de nature sismique ou de nature structurale, le format de stockage de cette information

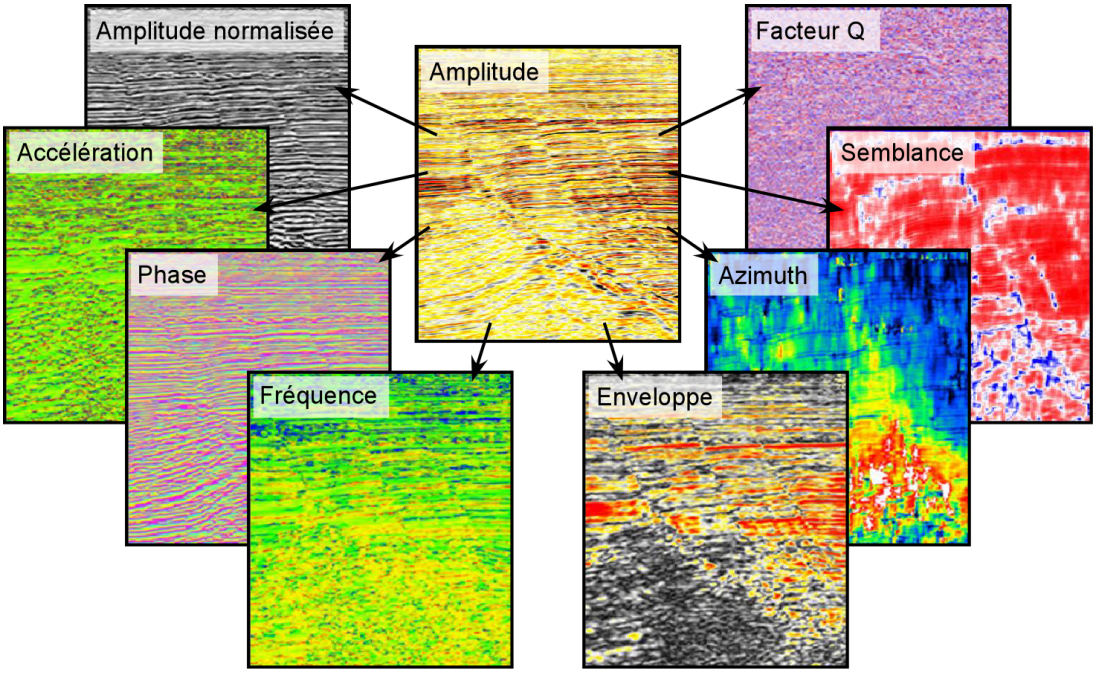

FIG. 4.1 – Exemples d'*attributs sismiques* dérivés du signal sismique initial par calculs basés sur le modèle mathématique de la *trace complexe* [TANER M.T. *et al.*, 1979].

est identique à celui du cube sismique, à savoir des grilles régulières. Dans ce cas précis, il est possible de substituer le terme *multi-attribut* au terme multimodal. Comme nous le verrons dans la suite de ce chapitre, les attributs sismiques, tout comme les cartes volumiques de distance, entrent dans cette catégorie d'information multimodale volumique.

Attributs sismiques

Nous avons précédemment évoqué la notion d'attribut sismique (Section 1.2.2), introduite à la fin des années 1970 [TANER M.T. et SHERIFF R.E., 1977] et définie par TANER, l'un des précurseurs dans le domaine, comme "toute information obtenue à partir des données sismiques, soit par mesure directe, soit par raisonnement logique ou basé sur l'expérience" [TANER M.T., 2001]. Le calcul d'attributs sismiques à partir du signal initial repose sur le modèle mathématique de la *trace complexe* [TANER M.T. *et al.*, 1979] et permet d'extraire des propriétés du signal sismique telles que la phase, la fréquence ou encore l'enveloppe (Figure 4.1). Celles-ci constituent des sources plus ou moins directes d'information géologique. D'autres attributs plus complexes comme la *semblance* permettent par exemple de mettre en évidence la présence de failles [BAHORICH M. et FARMER S., 1995]. D'une manière générale, tous ces attributs volumiques sont calculés à la résolution des données sismiques initiales.

Cartes volumiques de distance

Outre les attributs dérivés du signal sismique, qui constituent une information de nature sismique supplémentaire, d'autres volumes d'information sont potentiellement calculés au cours du processus d'interprétation. Par exemple, des objets géologiques extraits sous la forme de surfaces triangulées ou de lignes polygonales servent dans certains cas de base au calcul de *cartes*

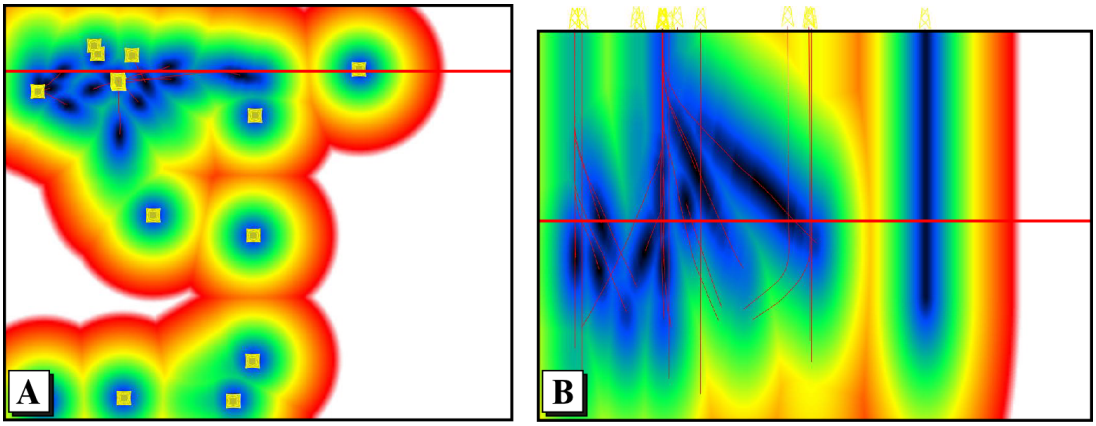

FIG. 4.2 – *Carte volumique de distance* construite à partir d'un ensemble de puits de production (derricks en jaune et tracés en rouge), et visualisée sur une coupe horizontale en profondeur (A) et sur une coupe verticale (B). En chaque point du volume est stockée la distance au tracé de puits le plus proche. Les lignes de coupe (A) et (B) apparaissent en rouge respectivement sur les coupes (B) et (A).

volumiques de distance. Il s'agit de grilles volumiques régulières qui, en chaque point, stockent la distance minimale à un ou plusieurs objects d'intérêt (Figure 4.2). De tels volumes d'information structurale viennent compléter les données sismiques initiales et les attributs qui en dérivent, permettant la visualisation des données sismiques dans un contexte structural particulier comme par exemple à une distance fixe d'une surface de faille ou d'un ensemble de puits.

Ainsi, le processus d'interprétation sismique ne se résume pas à la seule prise en compte du signal sismique initial mais plutôt à la combinaison de toutes ces sources d'information afin de converger plus rapidement vers une solution [MARSH A.J. *et al.*, 2000, CHRISTIE M., 2002]. Nous verrons en Section 4.2 les détails du système de visualisation volumique multimodale que nous avons mis en œuvre pour les besoins de l'interprétation sismique. Ce système repose sur un nouveau mode d'interaction avec l'utilisateur (calques volumiques), implanté en exploitant les capacités du matériel graphique moderne.

4.1.2 Les horizons sismiques comme source d'information multimodale

L'un des objectifs majeurs de l'interprétation sismique, notamment lors de l'analyse structurale, est la mise en évidence des principaux réflecteurs puis l'extraction sous forme surfacique des discontinuités géologiques correspondantes (Section 1.2.2). Comme nous avons pu le voir, ces discontinuités portent le nom d'horizons sismiques. Outre leur intérêt direct pour la construction du modèle structural dans la chaîne de modélisation en exploration-production (Section 1.2.1), ces horizons interviennent comme outils de contrainte structurale pour la visualisation des données sismiques. En effet, les coupes orthogonales classiques que sont les inlines, crosslines et coupes temps/profondeur (Section 1.3) présentent un certain nombre de limitations, notamment pour l'analyse stratigraphique, du fait qu'elles s'alignent sur les axes du volume sans tenir compte de l'orientation des structures d'intérêt. Ainsi, les coupes horizon qui ont été présentées en Section 1.3 permettent de visualiser les données sismiques à distance fixe d'un horizon précédemment extrait. L'intérêt d'une telle approche est qu'il est généralement difficile d'observer un événement sismique particulier tel qu'un chenal de manière continue par manque de repères

pour l'extraction de l'information. Comme nous l'avons montré sur la Figure 1.7, la présence d'un horizon à proximité fournit un tel repère.

Cette ré-injection d'information structurale dans le processus d'interprétation pour permettre d'affiner l'analyse peut véritablement être considérée comme une démarche de visualisation multimodale. Cependant, d'un point de vue pratique, le domaine de définition de l'information structurale impliquée est différent de celui des données sismiques visualisées dans la mesure où il s'agit de surfaces remarquables extraites du volume initial. Nous considérons donc qu'il s'agit d'un cas particulier de visualisation multimodale qui contraint la visualisation d'une information volumique par l'existence d'une information surfacique. Un système spécialisé pour ce type de scénario a été mis au point et sera présenté en Section 4.3. L'objectif est de permettre de visualiser les coupes horizon en perspective dans le volume d'information initial tout en assurant la mise à jour en temps réel de l'information volumique sur la coupe lors de variations de la distance de projection. Bien que relativement simple d'un point de vue technique pour des cubes sismiques de petite taille, la Section 4.3 montrera que la mise en œuvre d'un tel système devient problématique lorsque la taille augmente.

4.2 Notre système générique de visualisation volumique multimodale

La visualisation des données volumiques multimodales est un problème important de la visualisation scientifique qui a donné lieu au développement de techniques spécifiques. Nous allons tout d'abord en présenter les grandes lignes, ce qui permettra d'introduire le système que nous avons mis en place pour la visualisation volumique multimodale. Basé sur la notion de calque, ce système générique fait preuve d'une grande flexibilité et permet de reproduire de nombreux scénarios de visualisation. Nous nous intéresserons plus particulièrement à l'un d'entre eux qui, à notre connaissance, n'a encore jamais été proposé dans la littérature. Par simple extension du système précédent, ce dernier permet de combiner plusieurs champs scalaires en plaquant un ou plusieurs attributs sur des surfaces d'isovaleur d'un autre attribut, sans nécessiter d'extraction préalable de ces dernières sous forme polygonale. Ceci assure un meilleur temps de réponse du système.

4.2.1 Principe de la visualisation volumique multimodale

Pour la visualisation volumique, l'expression *vecteur multivarié* renvoie à la présence de plusieurs valeurs scalaires en chaque point d'un volume [Kaufman A. et Mueller K., 2005]. Ces vecteurs peuvent être l'expression de quantités physiques telles qu'une force ou une vitesse, ou la simple coexistence de plusieurs attributs sans relation directe. De nombreuses techniques ont été mises au point pour le cas de champs de vecteurs physiques, la plus connue étant certainement *LIC* (de l'anglais "Line Integral Convolution") [Cabral B. et Leedom L., 1993]. Nous nous intéressons cependant ici au cas de vecteurs d'attributs (par exemple les données sismiques).

Les techniques de visualisation de vecteurs d'attributs ont initialement vu le jour pour les besoins de la médecine. Dans ce contexte une fois de plus, deux scénarios sont envisageables. Les données dites *multicanales* correspondent à la définition de plusieurs canaux de couleurs, par exemple *RGB*, et sont entre autres obtenues par cryosections. Le projet du Visible Human [U.S. National Library of Medicine, –] en est un exemple. La visualisation de ces données présente l'avantage de ne pas nécessiter le recours à une fonction de transfert pour l'émission

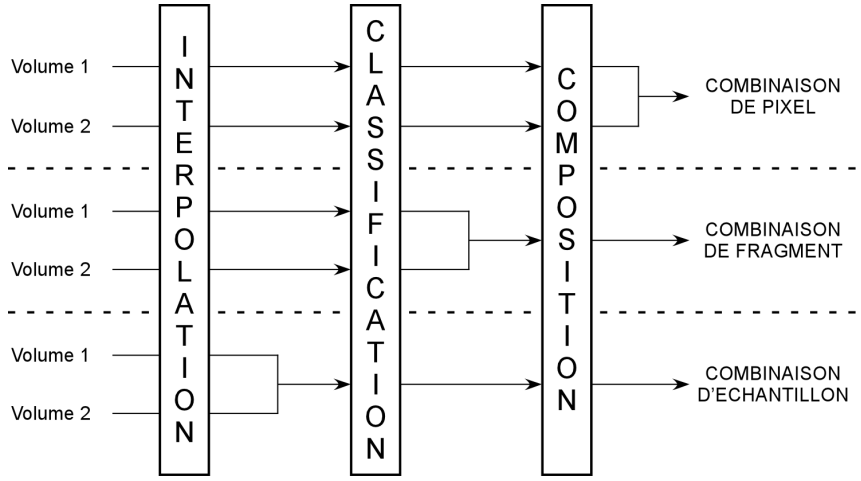

FIG. 4.3 – Des trois zones de combinaison potentielles en rendu volumique multimodal découlent trois niveaux de combinaison de l'information dans l'image finale : *combinaison de pixel*, *combinaison de fragment* et *combinaison d'échantillon*.

dans la mesure où les couleurs réelles sont disponibles. Il est cependant particulièrement difficile d'attribuer des valeurs d'opacité pour appliquer un modèle de rendu volumique tel que le modèle d'émission-absorption. En complément de méthodes semi-automatiques, l'introduction de fonctions de transfert multidimensionnelles peut alors s'avérer nécessaire [KNISS J. *et al.*, 2001a, KNISS J. *et al.*, 2002a]. Ce cas de figure ne présente qu'un intérêt limité dans le contexte des géosciences étant donné que l'obtention de coupes en couleurs réelles des objets à visualiser est peu probable, à l'exception des carottes de forage. Le second scénario concerne la présence de valeurs obtenues par des mesures différentes, telles que des mesures de tomographie et d'imagerie par résonance magnétique. Les données sont dites multimodales et c'est précisément ce cas qui nous intéresse.

Rendu volumique multimodal

La visualisation de données multimodales suppose l'existence dans l'algorithme de visualisation d'au moins une étape de combinaison des différentes sources d'information. Dans une méthode de visualisation volumique directe telle que le rendu volumique, il en existe potentiellement trois dont découlent trois niveaux de combinaison de l'information dans l'image finale [CAI W. et SAKAS G., 1999]. En effet, le calcul de l'intégrale de rendu volumique (Section 2.2.2) peut être décomposé en trois étapes majeures : interpolation, classification et composition. L'étape d'interpolation consiste à échantillonner le champ scalaire en interpolant les valeurs aux nœuds de la grille. À partir de cet échantillon, l'étape de classification associe des valeurs d'émission et d'opacité. En pratique, l'étape de classification sur le matériel graphique standard manipule des fragments. Enfin, pour chaque pixel de l'image finale, la contribution des fragments correspondants est accumulée par l'étape de composition. Comme le montre la Figure 4.3 pour le cas de deux volumes d'information, chacune de ces étapes peut être suivie d'une opération de combinaison qui produit une image différente. Dans la *combinaison de pixel*, l'image finale est obtenue par combinaison des images produites indépendamment pour chaque volume. Simple à mettre en œuvre dans la mesure où elle n'induit pas de modification dans l'algorithme de rendu volumique, cette technique présente l'inconvénient de filtrer l'information de profondeur.

La *combinaison de fragment* combine pour sa part l'émission et l'opacité obtenue indépendamment pour chaque volume au niveau du fragment, ce qui permet de conserver l'information de profondeur. Finalement, la *combinaison d'échantillon* combine les valeurs scalaires, par simple moyenne ou de manière plus évoluée, puis attribue une seule valeur d'émission et d'opacité par fragment en fonction de la valeur résultante.

À partir de cette classification générale des techniques de rendu volumique multimodal, il est possible de mettre en œuvre de nombreuses méthodes de combinaison de champs scalaires. Par exemple, Kniss *et al.* proposent l'utilisation de fonctions de transfert multidimensionnelles [Kniss J. *et al.*, 2001a, Kniss J. *et al.*, 2002a]. Ainsi, pour deux volumes, une interface évoluée d'édition de fonctions de transfert 2D permet d'affecter une valeur d'émission et d'opacité en fonction du couple de valeurs scalaires échantillonnées. Il s'agit en fait d'un cas particulier de combinaison d'échantillon reposant sur une combinaison évoluée des valeurs scalaires. À la limite, le rendu volumique préintégré qui a recours à une table 2D à l'étape de classification (Section 3.2.2), peut être considéré comme une forme de visualisation multimodale combinant deux volumes identiques. L'un aurait subi une légère translation par rapport à l'autre dans la direction d'observation de telle sorte qu'en chaque point un volume stocke la valeur en entrée de segment et l'autre la valeur en sortie. La technique des *masques volumétriques* (*"volumetric clipping"* en anglais) [Weiskopf D. *et al.*, 2003] est une autre méthode populaire de rendu volumique multimodal. Dans ce cas, un volume sert de masque d'opacité tandis que l'autre fournit l'émission. Il est ainsi possible de définir des formes arbitraires dans le masque qui vont permettre d'éliminer les parties moins intéressantes de l'autre volume. Cette technique illustre les possibilités de la combinaison de fragment. Dans le domaine de la visualisation des données sismiques, il est intéressant de remarquer que l'algorithme de "splatting" modifié présenté dans [Gerhardt A. *et al.*, 1999] qui fait intervenir une propriété d'appartenance à une région préalablement segmentée (Section 3.1.2) applique en fait un masque et peut à ce titre être vu comme une forme de combinaison de fragment.

Malgré ces quelques techniques mises en œuvre de manière ponctuelle pour des besoins particuliers, aucun effort à notre connaissance n'a été fourni pour proposer un système générique de visualisation volumique multimodale qui constituerait un cadre de travail général. C'est ce à quoi nous nous sommes attachés et les détails d'un tel système seront présentés à la Section 4.2.2.

Visualisation volumique multimodale à base de surfaces d'isovaleur

Face à la nécessité d'interpréter des données multimodales, un scénario de visualisation récurrent quel que soit le domaine d'application est le recours à des surfaces d'isovaleur. En effet, dans le cas simple de deux champs scalaires par exemple, la tendance naturelle conduit à plaquer le résultat de classification d'un champ sur des surfaces d'isovaleur d'un autre champ. Ceci permet par exemple de mettre en évidence la corrélation qui existe entre les deux champs. D'autre part, dans le cas où un champ correspond à une information géométrique, cette approche fournit un support de visualisation pour le second champ.

Ainsi, dans le domaine médical, Jen *et al.* dans [Jen D. *et al.*, 2004] utilisent cette technique de plaquage d'information sur des surfaces d'isovaleur pour l'étude des variations de la concentration d'une protéine le long de la membrane d'une dendrite de neurone. De même, en géosciences Frank a recours à des surfaces d'isovaleur d'une carte volumique de distance à un ensemble de puits pour contraindre la visualisation de champs scalaires volumiques [Frank T., 2006].

Cependant, la principale limitation des techniques existantes de visualisation multimodale à base de surfaces d'isovaleur est qu'à notre connaissance elles procèdent toutes de manière indirecte, à savoir par extraction des surfaces du premier volume suivie de leur rendu par le matériel graphique avec lecture des valeurs du champ scalaire dans le second volume. Cette approche par extraction préalable d'une représentation polygonale présente toutes les limitations des techniques de visualisation volumique par méthodes indirectes (Section 2.2.1), notamment la nécessité de procéder à une étape d'extraction dès que l'utilisateur modifie l'isovaleur, ce qui se traduit par un délai de mise à jour empêchant une utilisation interactive. Nous montrerons à la Section 4.2.3 comment, par une technique de combinaison de fragment, le système générique de visualisation volumique multimodale que nous avons mis en place permet de plaquer l'information de couleur d'un champ scalaire sur des surfaces d'isovaleur d'un autre champ scalaire sans en extraire de représentation polygonale, ce qui assure une mise à jour en temps réel pour toute modification de l'isovaleur.

4.2.2 Mise en œuvre d'une interface multimodale générique

L'objectif du système que nous avons mis en place pour la visualisation de données volumiques multimodales est de fournir un maximum de flexibilité à l'utilisateur. S'inspirant de l'interface des applications de dessin 2D, il repose sur la notion de calque dont l'extension au cas des volumes sera développée avant d'en présenter les aspects pratiques d'implantation.

Notion de calque

Dans la plupart des applications de dessin 2D, il est possible de définir un certain nombre de calques éditables de manière indépendante. L'image 2D produite par l'application est le résultat de la superposition dans une "troisième dimension" de ces calques indépendants. S'inspirant de cette approche, nous avons mis au point un système de visualisation volumique à base de calques 3D. De la même manière que les calques 2D des applications de dessin sont empilés dans la troisième dimension, ces calques volumiques sont empilés dans une "quatrième dimension" abstraite. Ainsi, en associant plusieurs volumes de données séparés à des calques différents, ce système fournit un moyen intuitif de les combiner.

L'édition indépendante de chaque calque repose sur la définition d'une fonction de transfert par calque. Suivant la nature du volume associé à un calque, la contribution désirée de ce dernier à l'image finale peut être différente. En effet, il est potentiellement préférable que certains volumes n'affectent que l'émission dans l'intégrale de rendu volumique et donc la couleur de l'image finale, tandis que d'autres ne doivent intervenir que dans les phénomènes d'absorption, n'affectant que l'opacité. Afin de simplifier la tâche de l'utilisateur dans cette démarche, nous avons défini trois types de calques en fonction du degré d'interaction dont il dispose avec la fonction de transfert (Figure 4.4) :

- **Couleur :** Les quatre canaux $RGBA$ sont modifiables dans la fonction de transfert. Il s'agit du type de calque qui expose la plus grande flexibilité, les canaux de couleur RGB pouvant être librement édités, tout comme le canal d'opacité A.
- **Opacité :** Les canaux RGB sont fixés à 1, ce qui produit indifféremment une couleur blanche, tandis que le canal d'opacité A peut être modifié. Ce type de calque permet par exemple d'affecter à un volume le rôle de masque.
- **Intensité :** Tandis que le canal d'opacité A est fixée à 1 sur tout l'espace de définition de la fonction de transfert, une couleur arbitraire est définie dont l'utilisateur module l'intensité

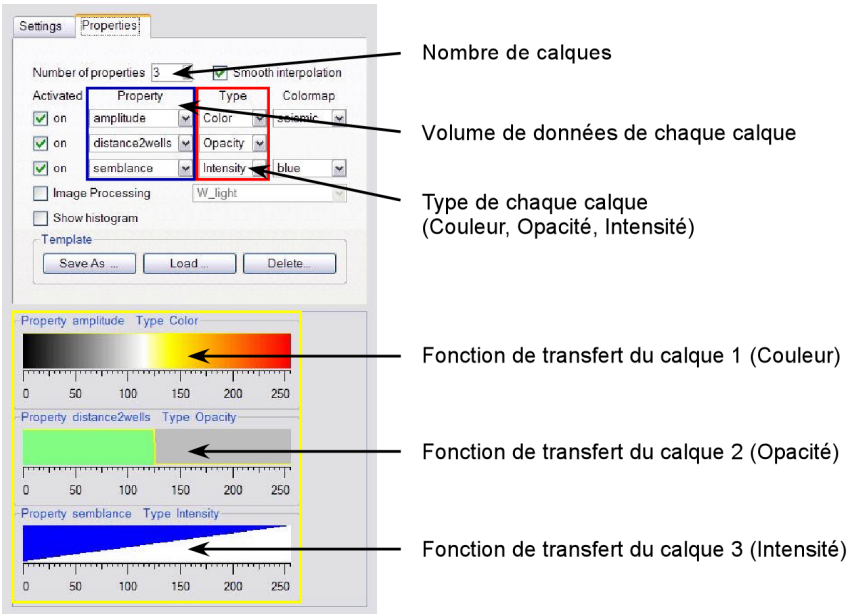

FIG. 4.4 – Interface d'édition de calques dans notre système de visualisation volumique multimodale. Trois calques ont été définis pour les volumes `amplitude`, `distance2wells` et `semblance`, respectivement de type *Couleur*, *Opacité* et *Intensité*. Les fonctions de transfert associées à chaque calque sont éditées indépendamment les unes des autres.

sur un fond blanc ou noir au choix. L'intérêt d'un tel calque est de simplifier le choix des couleurs pour un champ scalaire dont l'utilisateur veut rapidement voir les variations dans le volume.

Comme le montre la Figure 4.4, l'interface d'édition que nous avons mise en place permet la définition d'un nombre arbitraire de calques. La section suivante montrera que ce nombre est cependant limité par le nombre d'unités de texture disponibles sur la carte graphique (Section 2.3.1). Un champ scalaire est ensuite associé à chaque calque dont l'utilisateur définit également le type. Finalement, la fonction de transfert peut être éditée en fonction du type choisi. Cette interface, mise en place dans le logiciel VolumeExplorer pour les sondes de type Volume, est suffisamment générique pour imaginer divers scénarios de visualisation multimodale tout en gardant la possibilité de basculer sur un mode de rendu haute qualité comme celui présenté au Chapitre 3 en n'activant qu'un seul calque.

Implantation

Indépendamment des types de calques impliqués, le principe de leur combinaison est identique. Nous avons généralisé l'algorithme de rendu volumique de telle sorte que chaque volume atteint et passe l'étape de classification de manière indépendante. Celle-ci est fondée sur les fonctions de transfert éditées par l'utilisateur pour chaque calque. Puis, préalablement à l'étape de composition, les émissions et opacités de chaque fragment sont multipliées composante par composante. Ainsi, les canaux de couleur RGB et le canal d'opacité A sont multipliés indépendamment les uns des autres. Le résultat est composé pour mettre à jour la couleur du pixel correspondant. Il s'agit donc d'une forme de combinaison de fragment (Section 4.2.1) qui, comme nous l'avons précédemment signalé, a l'avantage de conserver l'information de profondeur. À noter cependant une différence notable par rapport aux systèmes de dessin 2D qui se contentent

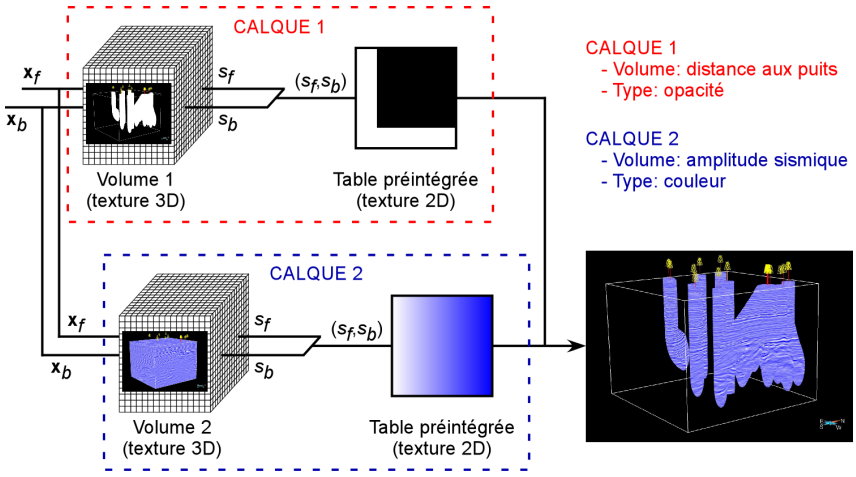

FIG. 4.5 – Combinaison de fragment sur deux calques volumiques avec classification préintégrée dans le programme de fragment pour le rendu volumique multimodal. Les paramètres en entrée sont les coordonnées de texture 3D des points sur les plans avant et arrière. Les résultats de la classification au niveau de chaque calque sont combinés pour produire la couleur finale du fragment.

d'une combinaison de pixel dans la mesure où la notion de fragment n'a pas de sens dans une image 2D monocouche.

Nous avons implanté cette combinaison de calques par combinaison de fragment à l'aide d'un programme de fragment qui applique la classification et effectue la multiplication canal par canal $RGBA$. Le choix du type de classification peut indifféremment se porter sur une simple post-classification, ou sur une classification plus élaborée avec préintégration (Section 3.2.2). La Figure 4.5 illustre la combinaison de deux calques avec classification préintégrée, où la lecture des émissions et opacités dans les tables préintégrées se fait de manière indépendante. Dans ce cas, le support de quatre unités de texture (Section 2.3.1) au moins est nécessaire, une texture 3D et une texture 2D étant utilisées pour chaque calque. D'une manière générale, la combinaison de n_c calques nécessite le support de $2n_c$ unités de texture. Bien que la plupart des cartes modernes en supportent seize, nous limitons volontairement le nombre de calques à cinq. En effet, les calques sont combinés deux à deux en une seule instruction peu coûteuse. Cependant, la classification préintégrée nécessite trois accès de texture (Section 3.3.1), lesquels sont particulièrement coûteux. Plus généralement, le nombre total d'instructions pour n_c calques est $4n_c - 1$, dont $3n_c$ accès de texture. Un scénario de post-classification réduit quelque peu ce nombre, chaque calque nécessitant deux accès de texture, l'un dans la texture 3D, l'autre dans une texture 1D qui stocke la fonction de transfert. Ainsi, le nombre total d'instructions est ramené à $3n_c - 1$ dont $2n_c$ accès de texture, ce qui reste élevé lorsque n_c dépasse cinq. D'autre part, pour éviter les exécutions conditionnelles qui ne sont pas supportées par toutes les cartes graphiques et qui peuvent s'avérer coûteuses, nous avons implanté plusieurs programmes de fragment utilisés en fonction du nombre de calques et du type de classification. Leur chargement dépend du scénario de visualisation courant. Encore significatifs il y a deux ans, il faut admettre que tous ces efforts paraissent aujourd'hui marginaux au vu des bandes passantes vers la mémoire de texture et de l'efficacité des caches sur les dernières générations de cartes graphiques. Toutefois, dans un contexte industriel, le délai de déploiement des toutes dernières technologies graphiques sur des parcs d'utilisateurs souvent conséquents est de l'ordre de plusieurs années, ce qui impose le support de configurations moins performantes.

Pour revenir brièvement sur la combinaison des calques par combinaison de pixel, il est intéressant de remarquer que, dans le cas de classification préintégrée, les fonctions de transfert sont préintégrées de manière indépendante. Au niveau des segments préintégrés, ceci revient en fait à effectuer de la combinaison de pixel et non de la combinaison de fragment au niveau des sous-segments de préintégration. En effet, une combinaison de fragment d'une telle granularité nécessiterait l'utilisation de tables préintégrées de dimension $2n_c$ pour tenir compte de tous les cas possibles de couples de valeurs en entrée et en sortie de tranche pour les n_c calques. Ceci n'est évidemment pas envisageable en pratique. Cependant, dans la mesure où la distance entre les plans reste raisonnablement faible, l'impact d'une combinaison de pixel au niveau des segments préintégrés est négligeable par rapport à la combinaison de fragment de l'image globale.

Exemple d'application : masques volumétriques

Tant du point de vue de son interface que de son implantation, ce système est particulièrement versatile et permet donc d'envisager des scénarios de visualisation multimodale extrêmement variés. Il est ainsi possible de reproduire la technique des masques volumétriques [Weiskopf D. *et al.*, 2003] initialement introduite de manière spécifique et appliquée à un cas particulier. Dans notre système, le volume servant de masque est simplement affecté à un calque de type Opacité tandis que l'autre est affecté à un calque de type Couleur. Il suffit ensuite d'éditer la courbe d'opacité du calque servant de masque pour éliminer les parties indésirables. La Figure 4.5 en donne un exemple où le rôle de masque est joué par une carte volumique de distance à un ensemble de puits tandis que la couleur dépend de l'amplitude sismique.

Le cas des masques volumétriques constitue un scénario parmi beaucoup d'autres possibilités. L'une des limitations d'une telle approche générique est la difficulté d'introduire des calculs d'illumination. En effet, il serait intéressant de pouvoir appliquer un modèle d'illumination locale sur la coupe introduite par le masque volumétrique [Weiskopf D. *et al.*, 2003]. Cependant, la force du système réside dans la généralité des programmes de fragment qui reposent sur le même moteur de rendu volumique et manipulent les calques indépendamment de leur type. L'illumination suppose le calcul de gradients sur l'un des champs scalaires et donc la spécialisation de ce dernier. Pour pallier à cette limitation, nous avons donc étendu le système à la visualisation d'un ou plusieurs champs scalaires par plaquage de leur information de couleur sur des surfaces d'isovaleur d'un autre champ, le tout avec calcul d'illumination. Cette extension repose sur le même moteur de rendu volumique et nécessite seulement le chargement d'un programme de fragment différent dans lequel l'un des champs est légèrement spécialisé.

4.2.3 Extension au plaquage d'information sur des surfaces d'isovaleur

La généralité du système qui vient d'être présenté en fait un candidat particulièrement intéressant pour le développement d'outils plus spécialisés et adaptés à un problème particulier. Un scénario répandu pour la visualisation volumique multimodale est la visualisation de surfaces d'isovaleur d'un champ scalaire avec plaquage d'une information de couleur provenant d'un autre champ (Section 4.2.1). La principale limitation des techniques existantes est l'extraction d'une représentation polygonale des surfaces par méthode de visualisation volumique indirecte. Cette représentation polygonale sert ensuite de support à la visualisation du second champ. Nous utilisons au contraire une technique de visualisation volumique directe pour le rendu de surfaces d'isovaleur initialement introduite pour les besoins de la visualisation de données médicales [Westermann R. et Ertl T., 1998]. Après en avoir exposé le principe, nous montrerons

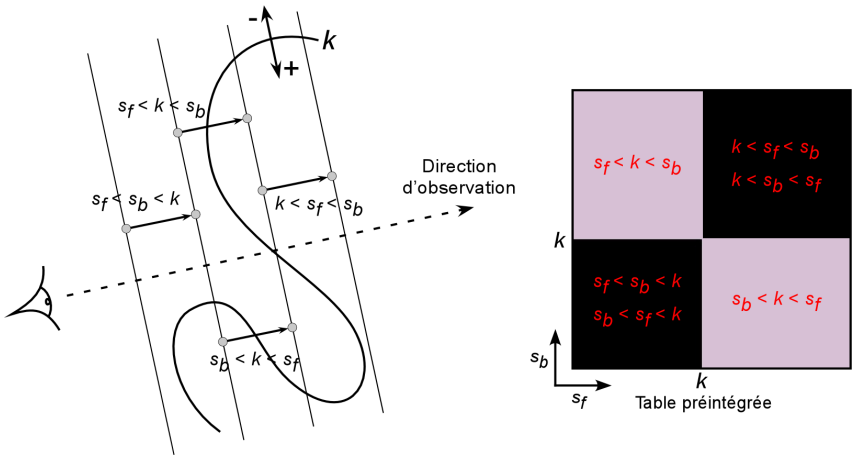

FIG. 4.6 – Cas particulier de table préintégrée pour la visualisation de surfaces d'isovaleur non polygonales. La surface d'isovaleur en gras est échantillonnée avec quatre tranches orthogonales à la direction d'observation. En fonction des valeurs du couple $\{s_f, s_b\}$ par rapport à l'isovaleur k, quatre cas distincts sont envisageables. Les rectangles noir et gris clair dans la table ont respectivement une opacité nulle et maximale.

comment cette approche originale permet de tirer directement parti du système de visualisation volumique multimodale que nous avons mis en place.

Surfaces d'isovaleur non polygonales

La principale limitation des techniques de visualisation volumique indirectes à base de surfaces d'isovaleur est le temps nécessaire à l'extraction préalable d'une représentation polygonale de la surface, extraction qui par définition n'intervient pas dans les techniques de visualisation volumique directes telles que le rendu volumique. Or, la description du rendu volumique en fait un candidat parfaitement adapté à la visualisation d'une surface équipotentielle d'un champ scalaire. En effet, l'utilisation d'une fonction de transfert qui assigne une opacité maximale pour une seule valeur k et une opacité nulle pour les autres permet de visualiser la surface d'isovaleur associée à k. L'ajout d'un modèle d'illumination locale de Phong (Section 3.2.1) fondé sur une estimation du gradient du champ scalaire produit des images de qualité satisfaisante. Cette forme particulière de surface d'isovaleur est dite non polygonale [WESTERMANN R. et ERTL T., 1998] du fait qu'aucune représentation polygonale n'est préalablement extraite. L'un des avantages majeurs est la possibilité d'éditer l'isovaleur k et d'observer le résultat en temps réel, à la seule condition de disposer d'un moteur de rendu volumique suffisamment performant.

Cas particulier de rendu volumique avec fonction de transfert adaptée, les surfaces d'isovaleur non polygonales sont donc également un cas particulier du rendu volumique préintégré avec table de lecture préintégrée spécifique [RÖTTGER S. *et al.*, 2000]. En effet, de manière intuitive il suffit de considérer qu'un segment $[s_f, s_b]$ intersecte la surface d'isovaleur si une seule de ses valeurs à l'extrémité s_f ou s_b est inférieure à l'isovaleur k, ou inversement si une seule des deux est supérieure à k. Dans ce cas l'opacité est maximale, sinon elle est nulle, ce qui donne une table en damier. Ceci est illustré sur la Figure 4.6 pour le cas du rendu volumique par méthode implicite à base de texture 3D dont la version préintégrée a été étendue pour la visualisation de surfaces d'isovaleur non polygonales [ENGEL K. *et al.*, 2001].

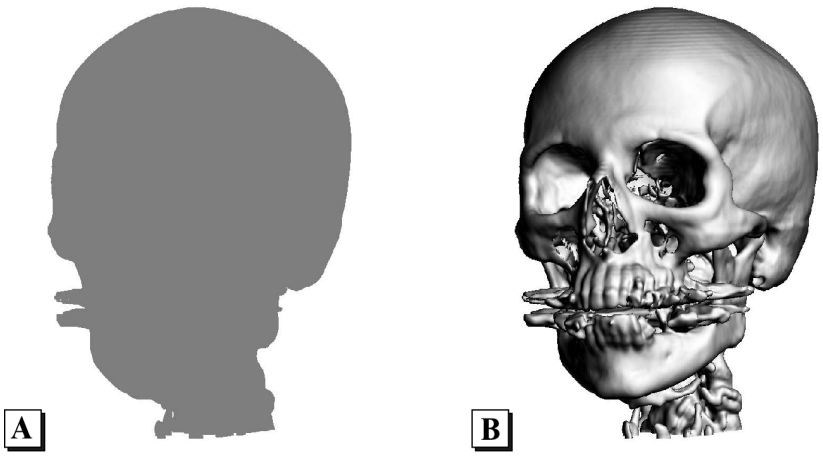

FIG. 4.7 – Visualisation d'une surface d'isovaleur non polygonale sur des données de scanner d'un crâne humain sans (A) et avec (B) application d'un modèle d'illumination locale.

Bien qu'initialement obtenue de manière intuitive, KRAUS et ERTL fournissent une démonstration rigoureuse de l'organisation en damier de la table dans leur étude du rendu volumique préintégré [KRAUS M. et ERTL T., 2005]. La fonction de transfert de l'opacité pour le cas de la visualisation d'une surface d'isovaleur k s'exprime comme suit :

$$\left\{ \begin{array}{rcll} \tau(s) & = & 0 & si\ s \neq k \\ \tau(k) & = & \infty & \end{array} \right. \tag{4.1}$$

ce qui revient à :

$$\tau(s) \ = \ \xi\ \delta(s-k) \quad | \quad \xi \in \mathbb{R} \tag{4.2}$$

où $\delta(x)$ est le delta de Dirac [WEISSTEIN E., –]. Dans ce contexte, l'expression de l'opacité d'un segment α_i de l'Équation (3.12) s'écrit comme suit :

$$\left| \begin{array}{l} \alpha_i \ = \ 1 - e^{-A_i} \\ \\ \text{avec : } A_i \ = \ \displaystyle\int_0^1 \xi\ \delta\Big((1-\omega)s_f + \omega s_b - k\Big)\ \Delta\ d\omega \end{array} \right. \tag{4.3}$$

où s_f et s_b sont respectivement les valeurs du champ scalaire en entrée et en sortie du segment, et Δ est sa longueur. En utilisant les propriétés du delta de Dirac suivantes [WEISSTEIN E., –] :

$$\left\{ \begin{array}{rcl} \delta(ax) & = & \dfrac{1}{|a|}\ \delta(x) \\ \\ \dfrac{d}{dx}H(x) & = & \delta(x) \end{array} \right. \tag{4.4}$$

où $H(x)$ est la distribution de Heaviside [WEISSTEIN E., –], le terme A_i se simplifie comme suit :

$$\begin{array}{rcl} A_i & = & \displaystyle\int_0^1 \xi \left| \frac{1}{s_b - s_f} \right| \delta\left(\omega - \frac{k - s_f}{s_b - s_f} \right) \Delta\ d\omega \\ \\ & = & \dfrac{\xi\ \Delta}{|s_b - s_f|}\ H\left(\dfrac{k - s_f}{s_b - s_f} \right)\ H\left(\dfrac{s_b - k}{s_b - s_f} \right) \end{array} \tag{4.5}$$

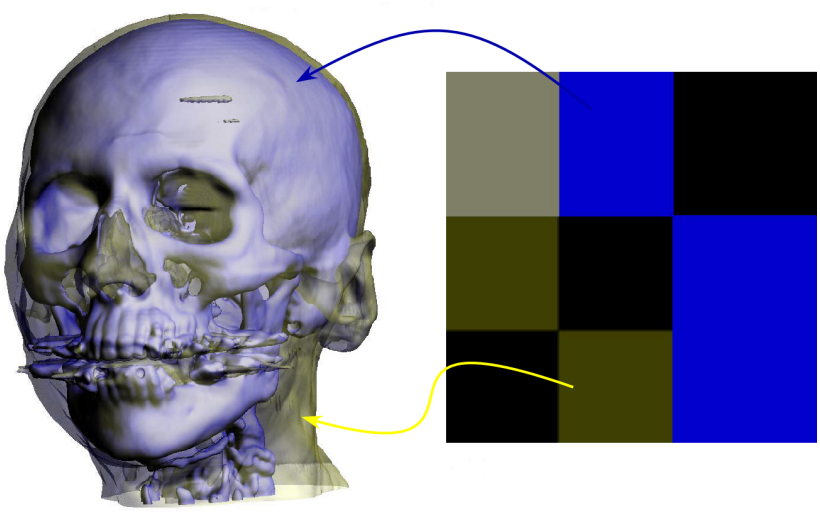

FIG. 4.8 – Visualisation de plusieurs surfaces d'isovaleur non polygonales semi-transparentes sur des données de scanner d'un crâne humain avec modèle d'illumination locale par simple adaptation de la table préintégrée.

Lorsque $\xi \rightarrow \infty$, l'égalité suivante est donc vérifiée :

$$\alpha_i \;=\; H\left(\frac{k - s_f}{s_b - s_f}\right) H\left(\frac{s_b - k}{s_b - s_f}\right) \tag{4.6}$$

expression de l'opacité α_i d'un segment ne dépendant que des valeurs du champ scalaire en entrée et en sortie, respectivement s_f et s_b, et qui, d'après les propriétés de la fonction de Heaviside, conduit effectivement à une table en damier telle que celle de la Figure 4.6.

Le choix de deux couleurs différentes pour les rectangles opaques de la table de la Figure 4.6 a pour effet de faire apparaître les deux faces de la surface avec des couleurs différentes. Cependant, les variations de couleur restent marginales tant qu'aucun modèle d'illumination locale n'est appliqué et il est impossible d'avoir une restitution correcte des effets de profondeur. Pour le calcul d'un modèle d'illumination locale précis, il est nécessaire de calculer le gradient en entrée et en sortie de tranche et d'interpoler les deux vecteurs pour obtenir le gradient sur la surface. Le coefficient d'interpolation dépend uniquement de s_f, s_b et k, mais son calcul sur le processeur de fragment surcharge inutilement ce dernier. Notre implantation des surfaces d'isovaleur non polygonales avec illumination locale utilise une table 2D précalculée qui, pour tout couple $\{s_f, s_b\}$, retourne la valeur du coefficient. Comme la table 2D préintégrée, celle-ci est mise à jour dès lors que l'utilisateur change la valeur de k. Elle est ensuite stockée sous forme de texture 2D en mémoire graphique et accédée par le programme de fragment chargé du calcul d'illumination. La Figure 4.7 montre le résultat de l'application d'un modèle d'illumination de Phong sur une surface d'isovaleur non polygonale. Sur l'image de la Figure 4.7-A produite sans calcul d'illumination, l'information de profondeur est absente tandis qu'elle est parfaitement restituée sur l'image de la Figure 4.7-B.

La technique s'étend très simplement à la visualisation de plusieurs surfaces d'isovaleur semi-transparentes. Seule la table de préintégration doit subir des modifications. Ainsi, la Figure 4.8 montre le résultat de la visualisation de deux surfaces d'isovaleur, l'une opaque l'autre semi-transparente, sur des données de scanner d'un crâne humain. La table préintégrée correspond à

la superposition des deux tables prises séparément en tenant compte des phénomènes d'occlusion éventuels lorsque la surface opaque se situe devant la surface semi-transparente. Inversement, l'apparition en transparence de la surface opaque lorsqu'elle se situe derrière est prise en compte. Dans le cas de plusieurs surfaces, nous appliquons les calculs d'illumination à la première surface visible sans tenir compte des effets éventuels de transparence.

Il important d'insister sur le fait que l'édition des isovaleurs n'a d'autre conséquence que la mise à jour de la table de préintégration (émission/opacité) et de celle des coefficients d'interpolation. Ces tables sont généralement dimensionnées en fonction de la taille de la fonction de transfert, soit 256^2 dans la grande majorité des cas (champ scalaire codé sur 8 bits). Sur notre Pentium M 1.69 GHz, ces deux calculs prennent seulement 0.001 secondes dans le cas de deux surfaces d'isovaleur. Ceci constitue un argument fort en faveur des surfaces d'isovaleur non polygonales par rapport aux techniques d'extraction dont l'étape de prétraitement est particulièrement coûteuse. D'autre part, la complexité algorithmique de l'algorithme de rendu reste indépendante du nombre de surfaces, puisque seule la table préintégrée 2D subit des modifications.

Visualisation volumique multimodale à base de surfaces d'isovaleur non polygonales

Comme nous venons de le voir, l'extension des techniques de rendu volumique par méthode implicite à base de texture 3D pour la visualisation de surfaces d'isovaleur non polygonales [Engel K. *et al.*, 2001] repose sur un principe similaire au rendu volumique préintégré par tranches épaisses. Or, ce dernier intervient comme élément de base pour le rendu de chacun des calques dans l'implantation de notre système de visualisation volumique multimodale (Section 4.2.2). L'intégration du concept de surface d'isovaleur non polygonale dans ce système est donc particulièrement naturelle. En effet, il suffit dans ce cas d'affecter à un des calques, le premier par exemple, le rôle de champ de surfaces d'isovaleur et d'intégrer le calcul du modèle d'illumination sur ce champ particulier. La contribution des autres champs reste définie par le type de calque auquel ils sont rattachés. Il devient ainsi possible de plaquer l'information de couleur issue d'un nombre arbitraire de champs sur les surfaces d'isovaleur d'un champ supplémentaire servant de support de visualisation. En effet, la multiplication indépendante des canaux $RGBA$ lors de la combinaison de fragment s'en charge de manière naturelle et automatique. La Figure 4.9 illustre la combinaison d'une source d'information structurale, sous la forme d'une carte volumique de distance à un ensemble de puits, avec une ou plusieurs sources d'information sismique, ici l'amplitude sismique et la vitesse de propagation des ondes acoustiques dans le sol, deux propriétés qui présentent des fréquences de variation suffisamment différentes pour pouvoir les visualiser conjointement de manière efficace. Ce type de visualisation s'avère particulièrement intéressant pour observer le comportement d'un ou plusieurs attributs de nature sismique dans un contexte géométrique et/ou structural particulier.

Le système de visualisation multimodale qui vient d'être présenté est destiné au cas de la combinaison de volumes d'information dont l'espace de définition est identique. Il a ainsi été possible de mettre en place un cadre de travail générique ouvrant la voie à de multiples scénarios de visualisation multimodale. Cependant, comme l'a démontré la Section 4.1.2, l'hypothèse d'un espace de définition unique n'est pas une généralité. L'effort de combinaison d'information va donc porter maintenant sur un cas particulier d'informations hétérogènes, à savoir la visualisation de volumes d'information sismique contrainte par des horizons sismiques, connu sous le nom de coupes horizon.

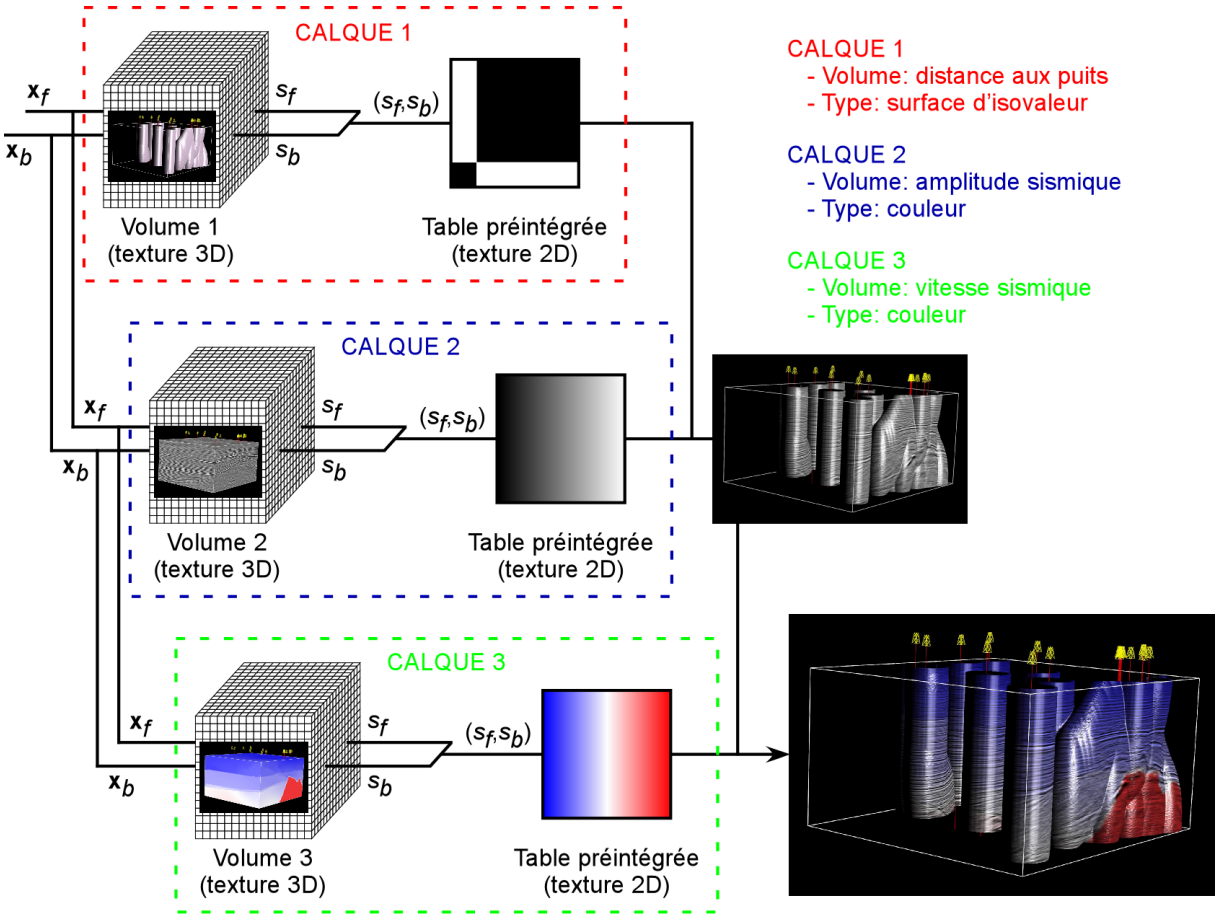

FIG. 4.9 – Visualisation volumique multimodale à base de surfaces d'isovaleur non polygonales. Le premier calque joue le rôle de surface d'isovaleur tandis qu'un nombre arbitraire de calques supplémentaires peuvent être combinés. Ainsi, l'utilisation de deux calques permet de plaquer les amplitudes sismiques en niveau de gris sur des surfaces d'équidistance à un ensemble de puits. La vitesse de propagation des ondes acoustiques dans le sol, propriété dont les fréquences de variations sont bien inférieures à l'amplitude sismique, peut également enrichir le rendu de l'image finale par l'ajout d'un calque supplémentaire.

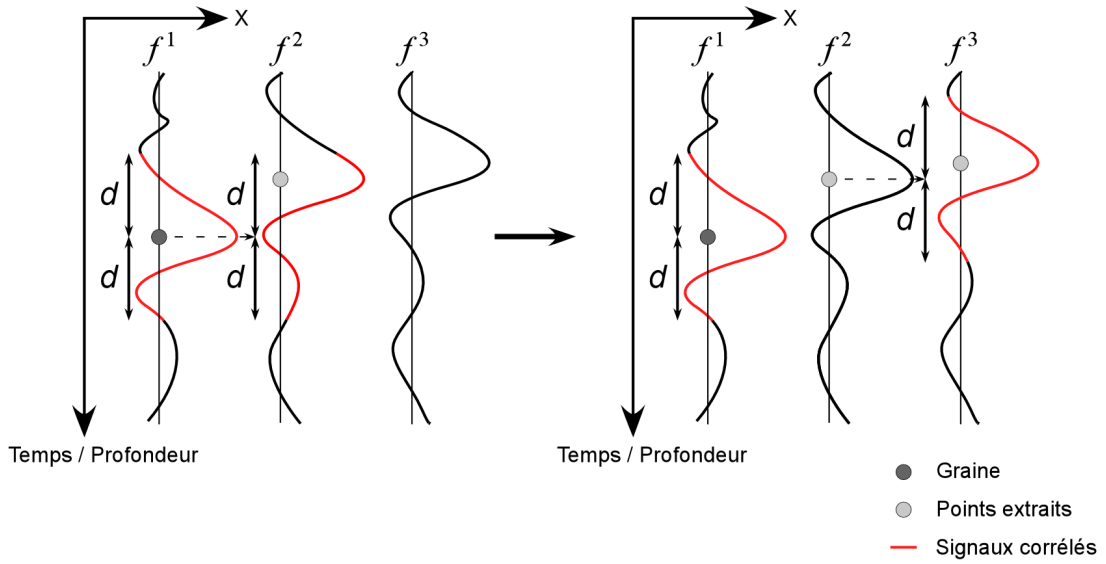

FIG. 4.10 – Principe de la propagation dans l'extraction semi-automatique des horizons sismiques. À partir d'une graine matérialisant la signature d'un horizon particulier sur une trace, l'algorithme se propage de trace en trace par corrélation des signaux dans un fenêtre verticale limitée (modifié d'après [LABRUNYE E., 2004]).

4.3 Un système spécialisé de visualisation multimodale à base d'horizons sismiques

Les horizons sismiques extraits au cours de l'interprétation constituent un important complément d'information structurale pour analyser les données sismiques de manière fine. En effet, les coupes horizon présentées en Section 1.3 permettent de sortir du schéma classique de coupes rectilignes alignées sur les axes du volume en fournissant un repère structural pour l'extraction d'information. Longtemps visualisées en 2D, l'apparition des stations graphiques au début des années 1990 a permis de les replacer dans leur contexte volumique en intégrant la perspective. Comme le montrera la Section 4.3.1, l'augmentation constante de la taille des volumes sismiques s'accompagne d'un certain nombre d'obstacles à la visualisation perspective en temps réel de ces coupes, même sur des systèmes graphiques modernes. Nous nous sommes donc intéressés aux techniques graphiques existantes pour la visualisation de ce type de surfaces afin de mettre au point un système de visualisation de grands horizons sismiques adapté à notre problème. Les détails de ce système feront l'objet de cette section.

4.3.1 Les horizons sismiques : extraction et représentation

L'alternance plus ou moins régulière dans le sous-sol de couches géologiques aux propriétés pétrophysiques différentes produit localement des contrastes d'impédance acoustique majeurs situés à la frontière entre ces couches. Ces limites géologiques, baptisées horizons géologiques, sont des réflecteurs privilégiés pour les ondes acoustiques et se matérialisent sur l'image sismique par des pics d'amplitude répétés latéralement sur un grand nombre de traces. L'extraction des horizons sismiques, expression sur l'image sismique des horizons géologiques, constitue donc un élément important dans la quête d'information structurale sur le sous-sol. La mise au point de méthodes semi-automatiques d'extraction a fait l'objet de nombreuses recherches depuis le

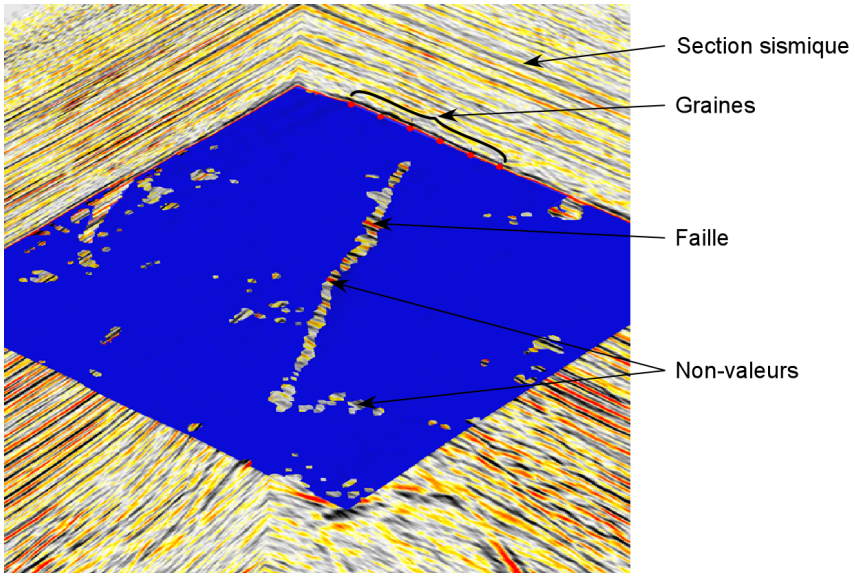

FIG. 4.11 – Exemple d'horizon sismique extrait à partir d'un ensemble de graines. La présence d'une faille affectant la continuité de l'horizon se caractérise pas une rupture de la propagation et produit des non-valeurs. D'autres zones de faible corrélation peuvent localement produire des non-valeurs (d'après [LABRUNYE E., 2004]).

début des années 1980 [KESKES N. *et al.*, 1983, FLINCHBAUGH F., 1986, HOWARD R.E., 1991]. Dans les méthodes modernes, un certain nombre de points correspondant à la signature sismique d'un même horizon sont sélectionnés sur quelques traces par l'utilisateur et servent de graines pour un algorithme de propagation automatique. Cet algorithme se propage de trace en trace par comparaison du signal sismique à la recherche de la signature initiale. Comme le montre la Figure 4.10, cette comparaison repose sur le calcul d'un coefficient de corrélation entre les traces voisines dans une fenêtre de taille verticale limitée. Pour éviter l'accumulation d'erreurs, le coefficient de corrélation est généralement calculé par rapport à la trace initiale.

L'algorithme d'extraction semi-automatique d'horizons du logiciel VolumeExplorer repose sur la projection du signal sismique sinusoïdal sur une base de polynômes trigonométriques [LABRUNYE E., 2004]. La fenêtre verticale de projection est définie sur des intervalles $[-w\delta t, +w\delta t]$ centrés sur la graine dans lesquels le signal f résultant s'écrit :

$$f(t) \;=\; a_0 \;+\; \sum_{k=1}^{m} \left(\; a_k \cdot \cos\!\left(\frac{k\pi t}{w\delta t}\right) \;+\; b_k \cdot \sin\!\left(\frac{k\pi t}{w\delta t}\right) \; \right) \tag{4.7}$$

où δt est la taille d'un voxel le long de l'axe t vertical, $a_0, \ldots, a_m$ et $b_1, \ldots, b_m$ sont les coefficients de la projection et m est le degré du polynôme ($m \leq w$, en pratique $m = w$ et $w = 10$). À partir de cette projection, il est possible de définir une fonction de corrélation R_{fg} entre deux polynômes trigonométriques f et g comme suit :

$$\forall\, \tau \in [-w\delta t, +w\delta t],\;\; R_{fg}(\tau) \;=\; \frac{C_{fg}(\tau)}{C_{ff}(0) \cdot C_{gg}(0)} \tag{4.8}$$

où C_{fg} est la covariance entre f et g définie par :

$$\forall\, \tau \in [-w\delta t, +w\delta t],\;\; C_{fg}(\tau) \;=\; \int_{-w\delta t}^{+w\delta t} f(t) \cdot g(t+\tau)\; dt \tag{4.9}$$

Le maximum de la fonction de corrélation R_{fg} sur la fenêtre $[-w\delta t, +w\delta t]$ définit le point de la trace voisine susceptible de correspondre à la signature recherchée. La propagation repose sur la définition par l'utilisateur d'une valeur seuil du coefficient de corrélation en dessous de laquelle la trace voisine est considérée comme dissemblable. La Figure 4.11 montre le résultat de l'extraction d'un horizon affecté par une faille qui produit une rupture de la propagation du fait de la faible corrélation qui résulte entre les traces voisines.

Les horizons ainsi extraits sont représentés de manière brute par un ensemble de points dans l'espace, chacun correspondant à la position de la signature sur une trace sismique. À chaque trace ne peut correspondre au plus qu'un point. Ainsi, le format de stockage qui s'impose naturellement est une grille régulière 2D de la résolution des coupes temps/profondeur du cube sismique dans laquelle est reportée la position des points le long de chaque trace. En informatique graphique, cette représentation particulièrement compacte porte le nom de *MNT*, pour *Modèle Numérique de Terrain*, ou *DEM* (de l'anglais "Digital Elevation Model"). Les appellations *carte d'élévation* (*"height field"* en anglais) ou tout simplement *terrain* sont également largement utilisées. Comme le montre la Figure 4.11, les horizons sismiques présentent cependant la particularité de contenir des non-valeurs sur les traces dont le coefficient de corrélation n'atteint pas le seuil fixé par l'utilisateur, phénomène qui n'existe pas dans les terrains manipulés en informatique graphique en général.

Le rendu surfacique de ces cartes d'élévation est obtenu en reliant quatre points voisins à l'aide de deux triangles. Cependant, lorsque la taille du cube sismique augmente, celle des horizons extraits suit une progression similaire. Ainsi, pour un volume de 4 GB par exemple dont les dimensions seraient 1000×2000×2000 (valeurs d'amplitude codées sur 8 bits), un horizon extrait peut potentiellement contenir 2000×2000 valeurs d'élévation, ce qui représente 8 millions de triangles à dessiner. Les cartes graphiques modernes parviennent à traiter environ 100 millions de triangles par seconde, ce qui permet de visualiser cette surface seule à environ 12 fps[13]. Tenant compte du fait que l'horizon n'a d'intérêt que s'il est visualisé dans son contexte, idéalement une sonde de rendu volumique, le taux de rafraîchissement de l'écran tombe rapidement sous la barre des 10 fps. Il est donc nécessaire de mettre au point des techniques spécifiques pour la visualisation de ces grands horizons sismiques. D'autre part, dans le contexte des coupes horizon, l'information volumique est projetée depuis une distance variable sur l'horizon ainsi visualisé. Cette projection repose sur l'extraction préalable des valeurs dans le cube à la distance spécifiée. Celles-ci sont alors stockées dans une texture 2D (Section 2.3.1) plaquée en temps réel sur l'horizon. Bien adaptée à de petits volumes, cette approche à base de texture 2D devient problématique dans le cas de volumes de grande taille pour lesquels l'extraction des données est particulièrement coûteuse et empêche l'édition en temps réel de la distance de projection par l'utilisateur.

En résumé, la visualisation de coupes horizon en perspective sur des volumes de grande taille avec édition en temps réel de la distance de projection se heurte à deux obstacles majeurs : la capacité de traitement de triangles limitée des cartes graphiques qui empêche le rendu en temps réel de la géométrie des horizons, ainsi que le temps d'extraction de l'information sismique dans le volume qui ne permet pas la mise à jour en temps réel de l'image lorsque l'utilisateur modifie la distance de projection. Comme nous le verrons à la section suivante, le premier problème est largement abordé dans la littérature et notre travail a consisté à rassembler un certain nombre d'idées existantes pour mettre en œuvre un système adapté à notre application dont les détails seront donnés en Section 4.3.3. Pour ce qui est de l'extraction d'information, le problème est

[13]de l'anglais "frames per second" (images par seconde)

très spécifique et la Section 4.3.4 présentera une projection en temps réel accélérée par la carte graphique en remplaçant la texture 2D par un ensemble de textures 3D.

4.3.2 La visualisation de terrains en informatique graphique

En informatique graphique, la visualisation de terrains de plusieurs millions voire dizaines de millions de polygones a fait l'objet de recherches intensives au cours des dix dernières années du fait de ses nombreuses applications, notamment dans les *Systèmes d'Information Géographiques* (*SIG*), les simulateurs de vol ou encore les jeux vidéo. Indépendamment de leur mise en œuvre, les techniques proposées ont pour objectif commun de réduire au maximum le nombre de polygones affichés à l'écran en affectant au minimum l'aspect visuel de la topographie du terrain. Cette réduction repose sur une hiérarchie de niveaux de raffinement permettant de sélectionner localement et en temps réel la résolution du maillage en fonction de divers paramètres qui minimisent l'erreur commise. La distinction entre toutes les techniques de visualisation de terrains proposées porte sur la structure de la hiérarchie de niveaux de raffinement utilisée.

Dans les *réseaux irréguliers triangulaires* ou *TINs* (de l'anglais "Triangular Irregular Networks) [Bonham-Carter G.F., 1994, Luebke D. *et al.*, 2003], un maillage 2D irrégulier est construit à partir de la carte d'élévation. Cette conversion repose selon les cas sur une triangulation de Delaunay [Schroder F. et Rossbach P., 1994, Cohen-Or D. et Levanoni Y., 1996, Cignoni P. *et al.*, 1997], ou bien sur une connectivité arbitraire [De Floriani L. *et al.*, 1997, Hoppe H., 1998, El-Sana J. et Varshney A., 1999]. Des techniques de multirésolution sont ensuite appliquées au maillage ainsi produit. Le principal intérêt des TINs est leur flexibilité. Cependant la topologie de la grille est définie de manière explicite, ce qui constitue un inconvénient majeur pour cette approche consommatrice de mémoire et lourde à mettre en œuvre.

Les *grilles régulières triangulées* conservent la régularité des cartes d'élévation qui définit implicitement la topologie par la position des sommets. Elles utilisent des *arbres binaires de triangles* (*"triangles bintree"* en anglais) [Lindstrom P. *et al.*, 1996, Duchaineau M.A. *et al.*, 1997, Pajarola R.B., 1998, Röttger S. *et al.*, 1998, Lindstrom P. et Pascucci V., 2002]. Les expressions *hiérarchie de triangles rectangles* ou *bissection de l'hypoténuse* (*"longest edge bissection"* en anglais) sont également rencontrées, deux noms qui illustrent bien cette approche consistant à subdiviser progressivement des triangles rectangles en insérant la médiane qui passe par l'hypoténuse. Ainsi, le niveau de raffinement est sélectionné localement lors du parcours en temps réel de l'arbre binaire en insérant successivement des arêtes.

Des approches hybrides permettent de tirer profit des avantages des deux camps en superposant une structure globale d'arbre binaire de triangles et des sous-ensembles de TINs [Evans W. *et al.*, 2001, Levenberg J., 2002, Cignoni P. *et al.*, 2003a, Cignoni P. *et al.*, 2003b]. Les feuilles de l'arbre binaire sont ainsi constituées d'ensembles triangulés irréguliers définis de manière générique.

Finalement, la dernière grande famille de méthodes repose sur une sélection par bloc des niveaux de détail [de Boer W., 2000, Larsen B. et Christensen N., 2003, Wagner D., 2004]. Dans ce cas, le terrain se voit imposer une structure globale d'*arbre quaternaire* (*"quadtree"* en anglais) [Samet H., 1984, Samet H., 1990] sur laquelle nous aurons l'occasion de revenir à la section suivante. Un parcours de l'arbre en temps réel de haut en bas permet de sélectionner rapidement le niveau de détail adéquat pour l'ensemble de triangles correspondant à une branche donnée. Dans [Lindstrom P. *et al.*, 1996], une sélection initiale par bloc est réalisée à partir d'un arbre quaternaire, puis une structure d'arbre binaire permet une sélection plus fine dans un second temps au niveau du bloc.

L'objectif premier de notre travail est la visualisation de données volumiques, la visualisation de terrain n'étant qu'un des moyens pour y parvenir. Aussi, une description exhaustive de l'ensemble de ces techniques dépasse le cadre de ce mémoire. La littérature en informatique graphique comporte de nombreux manuels traitant du sujet, notamment ceux destinés à la programmation de jeux vidéo. Le lecteur intéressé pourra se reporter à [Luebke D. *et al.*, 2003] ou [Terrain LOD, –] pour plus d'informations.

Il est cependant important dans le cadre de notre travail de citer la technique des *"Geometry Clipmaps"* [Losasso F. et Hoppe H., 2004], une méthode originale récente qui n'entre précisément dans aucune des familles précédemment citées. Cette approche repose sur une hiérarchie de grilles régulières 2D emboîtées et centrées sur le point de vue. Ces grilles d'extension croissante correspondent à des niveaux de détail incrémentaux. De cette manière, le niveau de détail est sélectionné en fonction de la distance au point de vue. Toute la difficulté de leur approche réside dans la mise à jour en temps réel de cette structure de grilles hiérarchiques lorsque l'utilisateur déplace le point de vue. Une implantation des "Geometry Clipmaps" a été proposée tirant partie des toutes dernières fonctionnalités des cartes graphiques pour déplacer au maximum le travail du CPU vers le GPU [Asirvatham A. et Hoppe H., 2005].

Dans le système que nous avons mis en place pour la visualisation de grands horizons (section suivante), nous nous sommes inspirés de plusieurs méthodes précédemment citées. Nous avons structuré les horizons sous formes d'arbres quaternaires, à l'image des techniques de sélection des niveaux de détail par bloc. Le parcours d'arbre est similaire à l'étape initiale de [Lindstrom P. *et al.*, 1996]. Cependant, la sélection des niveaux de détail repose sur le concept de hiérarchie de raffinement centrée sur l'utilisateur rencontré dans les "Geometry Clipmaps". D'autre part, un autre élément particulièrement important des "Geometry Clipmaps" dont nous nous inspirons est l'utilisation de la technique de *"displacement mapping"* en fonction du point de vue pour assurer une cohérence dans le temps et dans l'espace des niveaux de détail. Nous aurons l'occasion de revenir en détail sur ce dernier point à la section suivante.

4.3.3 Mise en œuvre d'une méthode de visualisation de grands horizons

À la lumière des techniques existantes de visualisation de terrains, un système a été mis en place pour la visualisation de grands horizons sismiques (plusieurs millions de triangles). La décimation géométrique en temps réel de la surface repose sur l'utilisation d'une structure d'arbre quaternaire. Cette décimation fait apparaître un certain nombre d'artefacts visuels que des techniques de programmation GPU permettent de réduire significativement. Après une description précise de la stratégie de décimation, les artefacts introduits seront clairement identifiés et les techniques qui permettent de s'en affranchir pourront être présentées.

Décimation géométrique en fonction du point de vue

L'objectif commun à toutes les techniques de visualisation de terrains est la réduction du nombre de triangles à afficher par la carte graphique. Dans notre système, cette décimation géométrique repose sur deux techniques complémentaires : l'*élimination des parties non visibles* et l'utilisation de *niveaux de détail*. La première est une technique conservative en ce sens que l'apparence de la surface n'est pas affectée, tandis que la seconde est non conservative, de légères modifications pouvant apparaître localement bien que l'apparence globale reste inchangée.

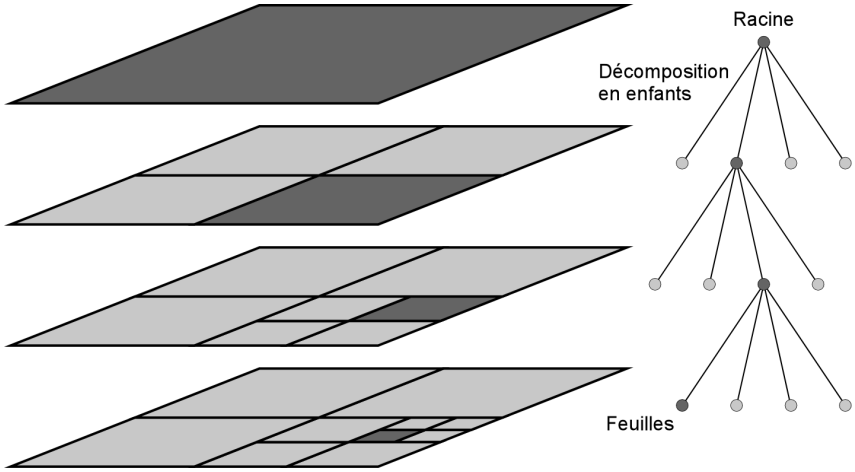

FIG. 4.12 – Principe de décomposition d'un horizon à l'aide d'un arbre quaternaire. Un nœud *racine* correspond à l'ensemble de l'horizon. Il est récursivement décomposé en quatre sous-ensembles réguliers, les nœuds *enfants*, jusqu'à une profondeur arbitraire. Les nœuds les plus profonds n'ont pas d'enfants et constituent les *feuilles* de l'arbre.

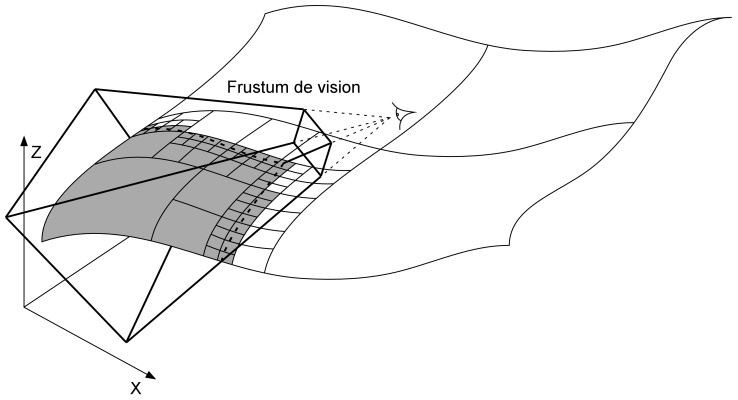

FIG. 4.13 – Élimination des parties non visibles basée sur une décomposition de l'horizon en arbre quaternaire. Les nœuds grisés correspondent à l'ensemble des nœuds visibles.

Élimination des parties non visibles Le champ de vision de l'observateur d'un monde virtuel en informatique graphique prend la forme d'une pyramide tronquée dans sa partie supérieure. Le terme correct est *frustum*, plus précisément *frustum de vision* dans ce contexte (Figure 4.13). Le frustum de vision est donc un sous-volume du monde virtuel hors duquel tout objet n'apparaît pas à l'observateur. En pratique, ce volume est défini par les six plans qui le délimitent. L'élimination des parties non visibles, communément appelée *"frustum culling"* en anglais, consiste à ne pas envoyer à la carte graphique les objets situés hors du frustum. Le calcul de visibilité repose sur six tests d'intersection avec les plans frontières. Le gain de performance résultant de cette approche dépend de la granularité à laquelle elle est appliquée. En effet, un raisonnement à l'échelle du triangle n'a pas d'intérêt dans la mesure où le gain de temps obtenu par la réduction du nombre de polygones dessinés par la carte graphique est perdu dans les calculs d'intersection sur le processeur central. À l'inverse, un raisonnement à l'échelle de la surface ne permettra pas de réduire le nombre de triangles.

C'est l'une des raisons pour lesquelles ont été introduites des structures de données de partition de l'espace. Ainsi, comme le montre le Figure 4.12 la décomposition d'un horizon en arbre quaternaire [Samet H., 1984, Samet H., 1990] regroupe progressivement les triangles par sous-ensembles logiques de l'espace depuis les *feuilles* de l'arbre jusqu'à sa *racine* qui correspond à l'horizon dans son ensemble. L'utilisation d'une telle structure de données hiérarchique autorise le calcul de visibilité à une échelle intermédiaire entre la surface et le triangle [Falby J.S. *et al.*, 1993, Lindstrom P. *et al.*, 1996]. L'algorithme parcourt l'arbre récursivement de haut en bas et remplit l'ensemble des nœuds visibles. Il est initialisé par un appel sur le nœud racine et s'écrit comme suit :

```
Visibilité_nœud
Entrée
   P : Nœud à tester
   F : Frustum de vision
Sortie
   V : Ensemble des nœuds visibles
Début
   Si {P ∩ F} ≠ ∅ // P est au moins partiellement visible
      Si P ⊂ F // P est entièrement visible
         Ajouter P à V
      Sinon // P intersecte la frontière du frustum
         Si P a des enfants
            Pour chaque enfant C de P
               Visibilité_nœud( C, F, V )
            Fin // Pour chaque
         Sinon // P est une feuille de l'arbre
            Ajouter P à V
         Fin // Si
      Fin // Si
   Fin // Si
Fin
```

La Figure 4.13 montre un exemple de calcul de visibilité par cet algorithme. Seuls les nœuds grisés sont visibles. Dans ce cas précis, le taux de décimation est de plus de 75% à partir de quelques calculs d'intersection seulement.

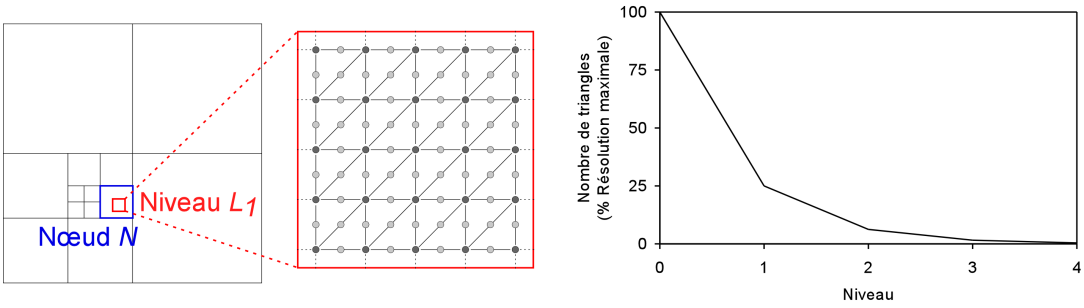

FIG. 4.14 – Effet de la décimation géométrique par niveau de détail sur un arbre quaternaire. L'agrandissement montre les sommets dessinés en gris sombre et les sommets correspondant à la résolution maximale en gris clair pour un nœud de l'arbre rendu au niveau de détail L_1. Le graphe donne la proportion de triangles dessinés à chaque niveau en pourcentage du nombre de triangles à la résolution maximale.

Cependant, le calcul de visibilité ne parvient pas à lui seul à assurer une réduction constante dans le temps du nombre de triangles dessinés par la carte graphique. En effet, lorsque l'utilisateur éloigne le point de vue pour avoir une vue globale de la scène, les performances se dégradent dans la mesure où un nombre croissant de triangles entrent dans le frustum de vision. Dans ce cas, le gain de performance passe nécessairement par des techniques non conservatives telles que l'introduction de niveaux de détail.

Niveaux de détail Dans le pire des cas où l'ensemble de la scène est contenu dans le frustum de vision, aucune accélération n'est observée par élimination des parties non visibles. Pourtant, dans un mode de vision en perspective, un éloignement du point de vue a pour effet de réduire la taille des polygones projetés sur l'écran, au point qu'elle devienne inférieure à celle d'un pixel et qu'apparaissent des phénomènes d'aliasage. Dans ce cas, le gain de performance peut être obtenu en décimant la surface sans pour autant en affecter significativement l'apparence. Nous définissons donc n niveaux de détail L_i ($i \in [0, n-1]$), où L_0 correspond à la résolution maximale et L_{n-1} à la plus faible résolution autorisée. Comme le montre la Figure 4.14 le rendu d'un nœud de l'arbre quaternaire au niveau de résolution L_i prend en compte un sommet sur 2^i. En conséquence, au niveau de détail L_i, un nœud contenant $(2^p + 1)^2$ sommets à résolution maximale sera rendu avec $2^{2(p-i)+1}$ triangles, ce qui divise le nombre de triangles entre chaque niveau par 2^2. Comme le montre le graphe, la plus grande partie de la décimation a lieu entre les niveaux L_0 et L_2.

S'inspirant de [FALBY J.S. *et al.*, 1993, LOSASSO F. et HOPPE H., 2004], notre stratégie de décimation est basée sur la distance au point de vue. $n-1$ sphères S_i ($i \in [1, n-1]$) de rayon r_i sont définies centrées sur le point de vue. Chaque sphère S_i correspond à la limite du niveau de détail L_{i-1}. Ainsi, un nœud de l'arbre contenu dans l'espace entre S_i et S_{i+1} est rendu au niveau de détail L_i. Entre le point de vue et S_1, il est rendu à la résolution maximale L_0 et au-delà de S_{n-1}, il est rendu au niveau L_{n-1}. Ces sphères emboîtées définissent donc une hiérarchie de niveaux de détail centrée sur le point de vue, tout comme les grilles régulières 2D des "Geometry Clipmaps". La sélection du niveau de détail se fait en temps réel par un parcours récursif de l'arbre de haut en bas. Tant qu'un nœud n'est pas entièrement contenu dans un niveau de détail, la sélection se propage en profondeur vers ses enfants jusqu'à ce qu'il ne soit plus possible d'affiner la recherche. Une attention particulière doit alors être apportée aux nœuds à cheval entre deux niveaux, dits *nœuds de transition* et traités de manière indépendante du reste de l'arbre. Dans

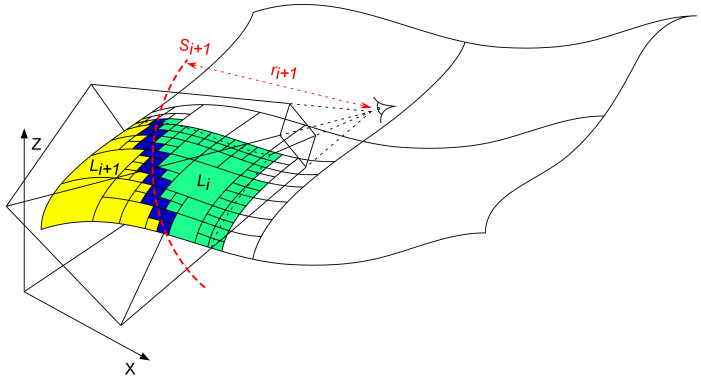

FIG. 4.15 – Sélection des niveaux de détail fondée sur une décomposition de l'horizon en arbre quaternaire. La sphère en pointillé rouge S_{i+1} définit la frontière entre les niveaux L_i et L_{i+1}. Les nœuds verts sont rendus au niveau de détail L_i et les nœuds jaunes au niveau L_{i+1}. Les nœuds bleus sont des *nœuds de transition* à cheval entre deux niveaux.

le cas contraire, des incohérences topologiques apparaissent sur lesquelles nous aurons l'occasion de revenir au paragraphe suivant. L'algorithme de sélection des niveaux de détail s'écrit comme suit :

Dessin_nœud
Entrée
P : Nœud à dessiner
Sortie
T : Ensemble des nœuds de transition (dessinés de manière différée)
Début
Pour chaque niveau de détail L_i (i de $n-1$ à 0)
Si $\{P \cap L_i\} \neq \emptyset$ // *P est au moins partiellement dans L_i*
Si $P \subset L_i$ // *P est entièrement dans L_i*
Dessiner P au niveau L_i
Sinon // *P intersecte la sphère S_i (à cheval sur L_{i-1} et L_i)*
Si P a des enfants
Pour chaque enfant C de P
Dessin_nœud(C, T)
Fin // *Pour chaque*
Sinon // *P est une feuille de l'arbre*
Ajouter P à T
Fin // *Si*
Fin // *Si*
Fin // *Si*
Fin // *Pour chaque*
Fin

La Figure 4.15 montre un exemple de sélection des niveaux de détail par cet algorithme sur l'ensemble des nœuds visibles calculés à la Figure 4.13.

Le rendu des nœuds utilise des *"triangle strips"*, fonctionnalité offerte par les librairies de programmation graphique telles qu'OpenGL (Section 2.3.1) pour minimiser le travail du processeur de sommets des cartes graphiques. D'autre part, deux autres fonctionnalités, les *tableaux de sommets* (*"vertex arrays"* en anglais) et les *buffers de sommets* (*"vertex buffers"* en anglais), permettent respectivement de réduire le nombre d'appels aux fonctions graphiques et de stocker

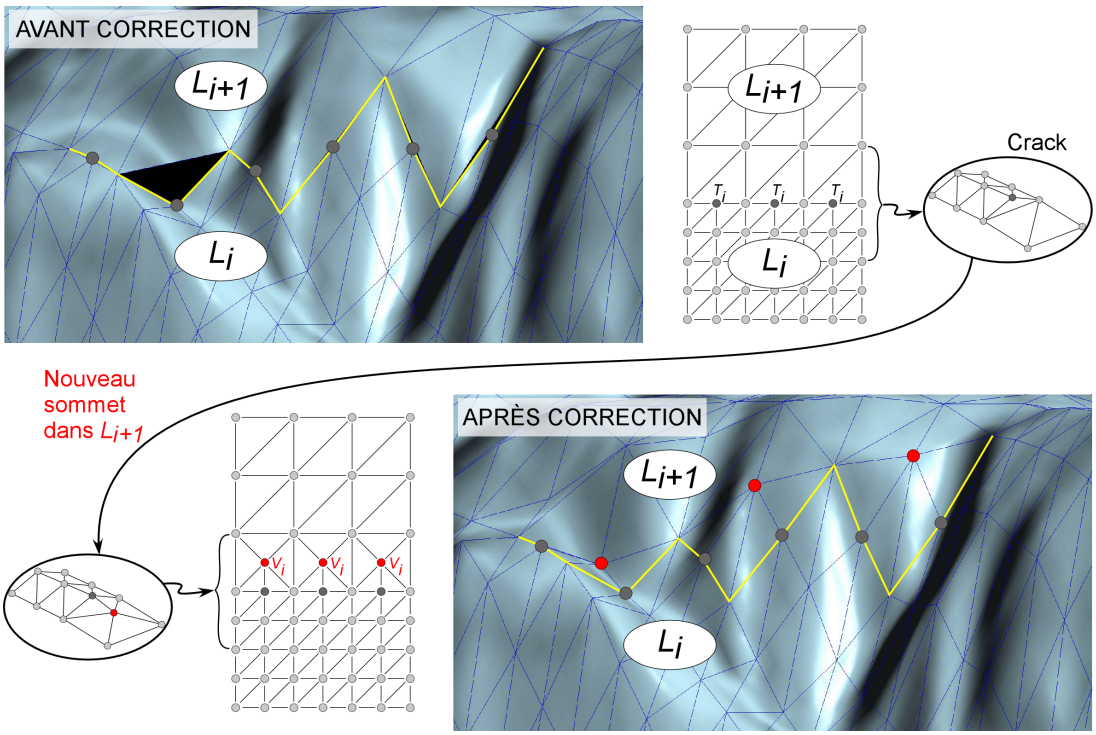

FIG. 4.16 – À la transition entre deux niveaux de détail (ligne jaune), la présence de sommets supplémentaires T_i (gris sombre) dans le niveau L_i génère des espaces vides. Ces singularités topologiques sont communément appelées *"cracks"*. L'ajout de sommets V_i (rouge) dans les nœuds de résolution inférieure L_{i+1} permet de s'en affranchir.

l'information sur les sommets directement dans la mémoire de la carte graphique. L'utilisation de "triangles strips" et de tableaux de sommets pour la visualisation de terrains remonte au début des années 2000 [DE BOER W., 2000], tandis que l'utilisation des buffers de sommets est plus récente [LOSASSO F. et HOPPE H., 2004].

Traitements topologiques L'introduction de niveaux de détail dans un rendu de surface nécessite d'apporter une attention particulière aux nœuds de transition entre deux niveaux où des singularités topologiques peuvent apparaître. Le terme employé est celui de *"cracks"* dont la bande supérieure de la Figure 4.16 donne des exemples. Ceux-ci résultent de la présence de sommets supplémentaires T_i dans le niveau de plus haute résolution. La tentation naturelle de déplacer ces sommets sur l'arête du nœud adjacent de résolution inférieure ne résout pas le problème. En effet, des erreurs d'arrondis numériques font clignoter certains pixels, phénomène qualifié de *jonction-T* (*"T-junction"* en anglais) [LUEBKE D. *et al.*, 2003]. Il est possible de s'affranchir des "cracks" et jonctions-T par l'ajout d'un triangle d'aire nulle au raccord entre les deux niveaux [LOSASSO F. et HOPPE H., 2004]. Nous utilisons cependant l'approche de [LARSEN B. et CHRISTENSEN N., 2003], illustrée dans la bande inférieure de la Figure 4.16, qui insère un sommet V_i au centre du nœud de résolution inférieure et dessine un éventail de triangles autour de ce sommet. Des techniques très similaires sont proposées dans [DE BOER W., 2000, WAGNER D., 2004]. Comme le montre la Figure 4.16, cette insertion permet de rétablir une topologie correcte.

L'application des deux techniques de décimation géométrique qui viennent d'être présentées permet de réduire significativement le nombre de polygones à dessiner tout en conservant une cohérence dans l'image produite (apparence globale de la surface conservée, pas d'espace vide, ...). Cependant la cohérence inter-image n'est pas assurée. En effet, durant la manipulation du point de vue par l'utilisateur, un nœud de l'arbre qui traverse une sphère S_i (i.e. passe d'un niveau de détail à un autre) subit une brusque modification géométrique du fait de l'apparation/disparition soudaine de sommets. Les artefacts ainsi produits portent le nom d'artefacts de *"popping"*[14]. Dans le cas des horizons sismiques, ces artefacts sont visuellement gênants lorsque les variations locales des valeurs d'altitude sont importantes. Il est possible de s'en affranchir sans affecter les performances par des techniques de programmation GPU.

Décimation et programmation GPU

La décimation géométrique telle qu'elle a été présentée s'accompagne d'un certain nombre d'artefacts. Les techniques de programmation GPU qui vont être décrites ont pour objectif d'en réduire l'impact visuel. Le *"geomorphing"*, initialement appliqué au cas des terrains par [Ferguson R.L. *et al.*, 1990], est plus particulièrement destiné aux artefacts de "popping" tandis que l'objectif de l'illumination par pixel [Blinn J.F., 1978] est d'appliquer un ombrage de Phong (Section 3.2.1) pour restituer la résolution initiale des calculs d'illumination ayant subi la même décimation que la géométrie.

"Geomorphing" Dans les méthodes de multirésolution, le "geomorphing" est une technique qui permet de restituer la cohérence inter-image par interpolation linéaire progressive de la géométrie des sommets entre les différents niveaux de détail. Comme le montre la Figure 4.17 cette interpolation fait intervenir un *facteur de "morphing"* m défini de manière indépendante pour chaque sommet de la grille. La géométrie du sommet résultant $\mathbf{v}$ est calculée comme suit :

$$\mathbf{v} \;=\; (1-m)\cdot\mathbf{v}_i \;+\; m\cdot\mathbf{v}_{i+1} \tag{4.10}$$

où $\mathbf{v}_i$ et $\mathbf{v}_{i+1}$ sont respectivement les positions du sommet aux niveaux de détail L_i et L_{i+1}. Cette interpolation n'a d'effet que sur les sommets qui disparaissent au niveau de détail L_{i+1} (ou inversement qui apparaissent au niveau de détail L_i), c'est-à-dire les sommets dont un des indices dans la grille régulière est impair (Figure 4.17). D'autre part, considérant que l'horizon est stocké dans une grille régulière, seule la valeur d'altitude est modifiée entre deux niveaux de détail, ce qui simplifie l'expression du "geomorphing" comme suit :

$$z \;=\; (1-m)\cdot z_i \;+\; m\cdot z_{i+1} \tag{4.11}$$

où z, z_i et z_{i+1} sont respectivement les altitudes du sommet interpolé, au niveau de détail L_i et au niveau de détail L_{i+1}.

Selon la manière dont le coefficient m est défini, il est possible de distinguer le "geomorphing" en temps de celui en espace. Dans le premier, l'interpolation occupe un laps de temps limité [Duchaineau M.A. *et al.*, 1997, Hoppe H., 1998, Larsen B. et Christensen N., 2003], ce qui suppose l'existence d'un mécanisme d'animation tenant à jour les temps de passage d'un niveau à l'autre. Dans le second, elle est fonction de la position des sommets par rapport au point de vue [Pajarola R.B., 1998, Röttger S. *et al.*, 1998, Cline D. et Egbert P.K., 2001,

[14] de l'anglais "to pop up" (surgir)

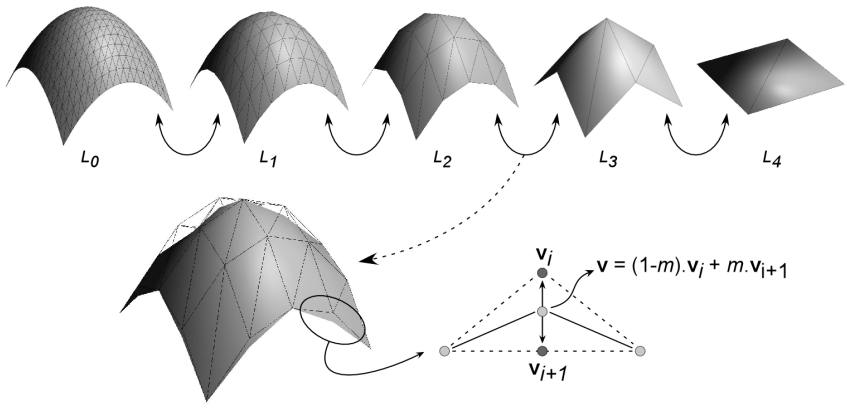

FIG. 4.17 – Principe d'interpolation progressive de la géométrie des sommets entre les niveaux de détail L_2 et L_3 ($i = 2$) par *"geomorphing"* sur une grille régulière de taille 17×17. Cette interpolation fait intervenir un *facteur de "morphing"* m défini par sommet.

WAGNER D., 2004, LOSASSO F. et HOPPE H., 2004]. Dans la mesure où notre sélection de niveaux de détail, inspirée de [LOSASSO F. et HOPPE H., 2004], dépend de cette même position, nous utilisons naturellement un "geomorphing" en espace. La position des sommets dépend ainsi exclusivement du point de vue, ce qui assure une cohérence du maillage dans le temps. En effet, pour une position de l'observateur dans l'espace donnée, celui-ci a toujours le même aspect. D'autre part, cette approche ne nécessite aucun mécanisme supplémentaire de conservation ni de mise à jour des temps d'initiation de l'interpolation. Pour un sommet appartenant au niveau de détail L_i et situé à la distance d de l'observateur, le facteur m est défini comme suit :

$$m \;=\; m_i(d) \;=\; \begin{cases} max(\ 2d/r_1 - 1,\ 0\ ) & i \;=\; 0 \\ (d - r_i)\ /\ (r_{i+1} - r_i) & i \;\in\; [1, n-2] \end{cases} \tag{4.12}$$

où r_i ($i \in [1, n-1]$) sont les rayons des sphères S_i introduites au paragraphe précédent. La zone d'interpolation ainsi définie est continue dans l'espace. En effet, un sommet n'apparaît à sa position L_i que sur la sphère S_i, à l'exception des niveaux L_0 et L_{n-1}. Cet entrelacement des niveaux de détail est légèrement différent de l'approche de [LOSASSO F. et HOPPE H., 2004] qui définit une zone de transition d'extension limitée.

Malgré la cohérence en temps et en espace des niveaux de détail qui résulte de l'introduction du "geomorphing", une implantation de la technique sur le CPU est particulièrement coûteuse dans la mesure où l'altitude de chaque sommet doit être modifiée avant l'envoi sur la carte graphique. Nous utilisons au contraire la technique de "displacement mapping" qui implante l'interpolation dans un programme de sommet (Section 2.3.1) de la même manière que [LARSEN B. et CHRISTENSEN N., 2003, WAGNER D., 2004, LOSASSO F. et HOPPE H., 2004]. Le programme de sommet calcule le facteur m et en déduit un vecteur de déplacement qui modifie la géométrie du sommet. L'altitude z_{i+1} au niveau de détail L_{i+1} d'un sommet apparaissant/disparraissant au niveau de détail L_i est précalculée et stockée de manière contiguë dans le même buffer de sommets que la géométrie initiale. De cette manière, les altitudes haute résolution z_i et basse résolution z_{i+1} sont accessibles par le programme de sommet pour calculer en temps réel la géométrie interpolée. L'augmentation du nombre n de niveaux de détail supportés augmente parallèlement la consommation de mémoire graphique, un $(n-1)$-vecteur supplémentaire devant être stocké pour chaque sommet. En pratique, nous utilisons cinq niveaux de détail : la géométrie initiale plus quatre niveaux de décimation, ce qui conduit au stockage d'un 4-vecteur

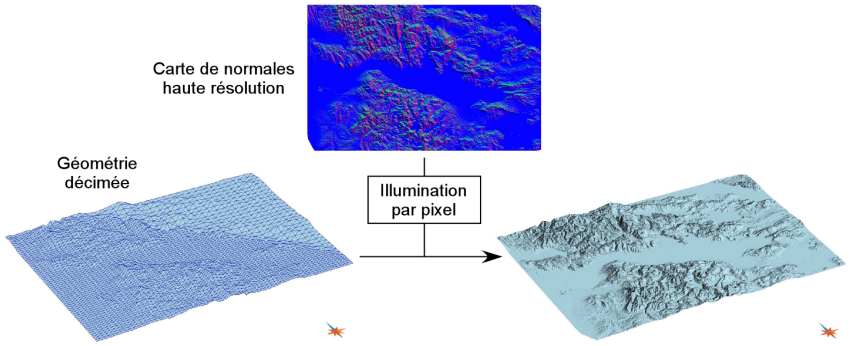

FIG. 4.18 – Principe de l'illumination par pixel sur un terrain décimé. La combinaison d'une géométrie basse résolution avec une *carte de normales* haute résolution permet de maintenir un niveau de performance correct tout en conservant des effets d'illumination détaillés (vue du Golfe de Corinthe, Grèce).

supplémentaire par sommet. Les attributs supplémentaires sur les sommets sont en effet stockés au minimum sur des 4-vecteurs. L'utilisation d'un seul niveau de détail supplémentaire obligerait donc à stocker un vecteur supplémentaire, tandis que l'utilisation d'un, deux ou trois niveaux en moins ne permettrait pas de réduire la consommation mémoire.

Illumination par pixel Comme nous l'avons vu en Section 2.3.1, les calculs d'illumination sur le matériel graphique standard sont exécutés par le processeur de sommet, un vecteur normal ayant été spécifié pour chaque sommet ainsi qu'un ensemble de paramètres d'éclairage globaux (position des sources lumineuses, couleur, ...). À l'étape de rasterisation, la couleur résultant de l'éclairage calculé par le processeur de sommet est interpolée bilinéairement dans chaque fragment, laquelle est éventuellement combinée ultérieurement avec une couleur de texture sur le processeur de fragment. Le résultat de cette implantation est un ombrage de Gouraud (Section 3.2.1), les calculs d'illumination ayant été effectués par sommet. Dans l'utilisation de niveaux de détail, la géométrie décimée envoyée à la carte graphique a pour conséquence de produire des effets d'éclairage basse résolution ayant subi la même décimation. Pour produire des effets d'illumination par pixel, il est nécessaire d'appliquer un ombrage de Phong en effectuant les calculs d'éclairage au niveau du fragment. Ce type d'illumination par pixel porte couramment le nom de *"bump mapping"* [BLINN J.F., 1978]. L'utilisation d'un programme de fragment sur les cartes graphiques modernes permet une implantation temps réel. Les normales à l'horizon sont stockées à la résolution initiale dans une texture 2D, la *carte de normales*. Au moment du rendu, le programme de fragment accède à cette texture et calcule l'éclairage pour chaque fragment de manière indépendante. Comme le montre la Figure 4.18, les effets d'illumination sont ainsi produits à l'échelle du pixel malgré la décimation géométrique de la surface.

D'autre part, du fait de la présence de failles, de dômes de sel ou de bruit dans les données, les horizons extraits en interprétation sismique contiennent souvent des non-valeurs (Figure 4.11). Dans les algorithmes de visualisation de terrains classiques, cette éventualité n'est pas envisagée dans la mesure où les terrains sont généralement continus. Nous gérons les non-valeurs à l'aide du canal d'opacité de la texture stockant la carte de normales. En effet, les textures sont généralement stockées dans un espace de couleur $RGBA$. Cependant, les normales sont définies par seulement trois composantes, stockées dans les canaux RGB de la texture. Nous utilisons donc le canal d'opacité A pour stocker un masque binaire égal à 0 dans les zones de non-valeurs et 1 partout ailleurs. En fonction de la valeur du masque, le programme de fragment génère un

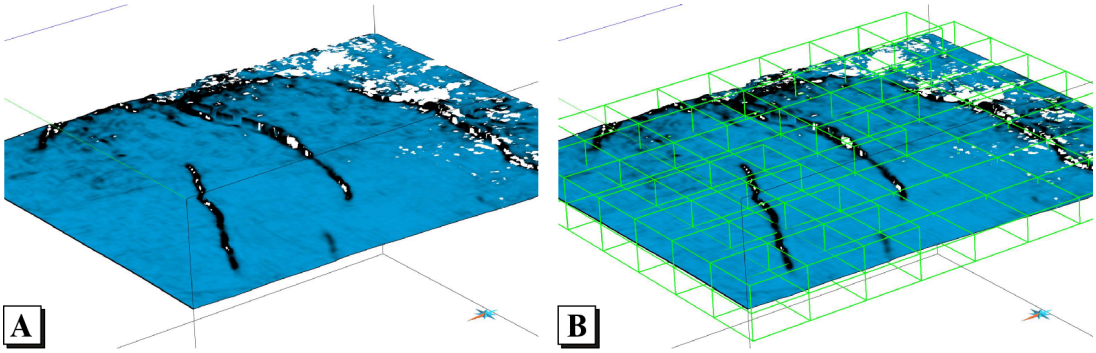

FIG. 4.19 – Visualisation de l'attribut de *semblance* sur une coupe horizon en perspective. La distance de projection est nulle et les faibles/fortes valeurs de semblance apparaissent en noir/bleu. (A) Projection à base de texture 2D. (B) Projection à base de texture 3D avec cache de texture (les briques résidantes sont représentées en vert).

fragment transparent ou opaque. Nous montrons Figure 4.19 un exemple d'horizon faillé avec un rendu correct des zones de non-valeurs.

La méthode de visualisation de grands horizons sismiques qui vient d'être présentée repose donc sur un certain nombre de techniques existantes pour la gestion de niveaux de détails sur un modèle numérique de terrain. Ces techniques, ayant fait l'objet d'études approfondies par le passé (Section 4.3.2), sont largement reconnues dans la communauté scientifique. Cependant, l'une des principales limitations de notre approche est l'absence de contrôle d'erreur. En effet, le niveau de décimation dépend uniquement de la position par rapport au point de vue. Bien qu'intuitivement correcte, cette stratégie peut conduire à des erreurs notables lorsque les sphères S_i ne sont pas correctement dimensionnées. Cependant, dans le cadre de notre application, la précision est un élément capital lorsque l'utilisateur effectue un agrandissement maximal pour observer l'horizon au contact d'une section du cube sismique par exemple, ce qui est assuré par la définition du facteur d'interpolation m de l'Équation (4.12). En effet, celle-ci assure un rendu à la résolution maximale L_0 dans toute la première moitié de la sphère S_1. D'autre part, lorsque l'utilisateur éloigne le point de vue, les erreurs sont en pratique peu visibles dans la mesure où les horizons sismiques sont des surfaces dont les fréquences de variation de l'altitude restent relativement faibles.

4.3.4 Projection de l'information sismique accélérée par la carte graphique

La méthode de visualisation de grands horizons sismiques qui vient d'être détaillée sert de base à un système de visualisation de coupes horizon en perspective. De la même manière qu'une texture 2D stocke la carte de normales pour les calculs d'illumination au niveau de chaque fragment, il est possible d'utiliser une texture 2D supplémentaire pour stocker une information sismique qui est ensuite plaquée sur l'horizon. Nous montrons une telle coupe horizon en perspective sur la Figure 4.19-A où l'information sismique projetée contient l'attribut de *semblance*, un attribut particulier qui met en évidence la présence de failles [TANER M.T. et KOEHLER, 1969]. Les faibles valeurs de semblance (noir) correspondent à une faible corrélation latérale des traces sismiques et inversement pour les fortes valeurs (bleu). Dans ce cas, la distance de projection est nulle, ce qui signifie que l'information plaquée correspond aux données extraites dans le cube à la position exacte de l'horizon. La Figure 1.7 a déjà montré les possibilités qu'offre la

modification de cette distance. Cependant, chaque modification implique une extraction de l'information sismique dans le cube pour la stocker dans une texture 2D. Or, dans le cas de volumes de grande taille, cette extraction qui produit des accès ponctuels dans le cube peut s'avérer particulièrement coûteuse.

Une solution plus adaptée dans ce cas est l'utilisation de textures 3D pour le stockage des données sismiques. L'information sismique peut alors être projetée en temps réel à l'aide de coordonnées de textures définies de manière adéquate. Nous utilisons la fonctionnalité de génération automatique de coordonnées de texture de la librairie OpenGL qui permet de définir ces dernières de manière implicite à partir de la position géométrique des sommets. Ainsi, une modification de la distance de projection est interprétée en termes de translation des coordonnées de texture le long du vecteur de projection, ce qui est effectué par une simple mise à jour des paramètres de génération automatique des coordonnées de texture. Cette mise à jour implique seulement quatre valeurs flottantes à envoyer à la carte graphique pour effectuer correctement les calculs ultérieurs de coordonnées, à comparer à l'extraction complète d'une nouvelle texture du cube dans le cas de l'utilisation de textures 2D.

L'inconvénient majeur de cette projection accélérée par la carte graphique par l'utilisation de textures 3D est la quantité de mémoire graphique nécessaire pour le stockage des textures. Celle-ci est limitée et dépasse rarement 512 Mo sur les cartes graphiques modernes. Notre implantation des textures 3D utilise une décomposition du volume en briques élémentaires qui peuvent être chargées séparément en mémoire graphique [Castanié L. *et al.*, 2005c]. Cette décomposition est à la base d'un système de cache qui fera l'objet du Chapitre 5. Comme nous le verrons, un ensemble de briques dites *résidantes*, chargées en mémoire de texture, est mis à jour en permanence de telle sorte que les données nécessaires à chaque instant soient chargées sans dépasser la quantité de mémoire disponible. Le Figure 4.19-B montre un ensemble de briques stockant l'attribut de semblance en mémoire de texture dans le voisinage d'une coupe horizon en perspective pour assurer une projection en temps réel de l'information sismique sur la géométrie de l'horizon.

4.4 Bilan et perspectives

Ce chapitre nous a donc permis d'exposer deux systèmes distincts de combinaison d'information dans le cadre de l'interprétation sismique : l'un est générique et concerne les cas d'information multimodale exclusivement volumique, l'autre est spécialisé au cas de la combinaison d'une ou plusieurs sources volumiques avec une source surfacique. Dans les deux cas, il est possible de mettre en relation l'information structurale disponible avec les données sismiques initiales ou dérivées du signal initial, tout ceci en temps réel et avec un maximum de flexibilité pour servir au mieux le processus d'interprétation.

Pour aller plus loin en ce qui concerne le cas particulier de plusieurs volumes d'informations, il pourrait être intéressant d'explorer l'utilisation d'outils de combinaison pondérée qui appliquent des coefficients multiplicateurs aux données d'émission/absorption issues de la classification. Cette technique utilisée dans [Frank T., 2006] permet de faire varier la contribution de chaque volume d'information au rendu final et semble intéressante pour mettre un attribut particulièrement important en avant. De même, nous avons évoqué l'existence de trois niveaux différents de combinaison de l'information en rendu volumique multimodal (Section 4.2.1). Bien que nous étant contentés de travailler sur la combinaison de fragment, il semble particulièrement intéressant de développer l'approche par combinaison d'échantillon (Figure 4.3) dans le

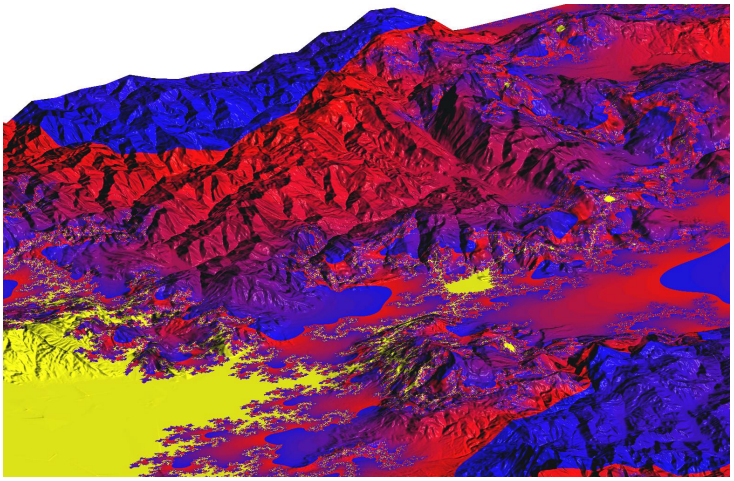

FIG. 4.20 – Calcul d'une fractale (ensemble de Mandelbrot) par pixel sur un terrain décimé (vue du Golfe de Corinthe, Grèce). L'équation caractérisant l'appartenance à l'ensemble de Mandelbrot est évaluée en temps réel par le processeur graphique dans un programme de fragment.

contexte de l'interprétation sismique dont bénéficierait notamment l'interprétation des données de *sismique 4D*. Ce cas particulier d'interprétation concerne la mise en relation de deux cubes sismiques correspondant à des campagnes décalées dans le temps. Ainsi, la comparaison de certains indicateurs d'hydrocarbures directs permet d'observer les phénomènes de migration de fluides dans le réservoir après plusieurs années de mise en production. Dans ce cas, la combinaison des valeurs d'émission/opacité après classification n'est pas d'un grand intérêt. Par contre, le calcul d'une valeur dérivée des deux volumes, par différence par exemple, pour appliquer une classification unique au résultat fournit un moyen efficace de déceler rapidement les zones du réservoir ayant subi de fortes migrations de fluides, laquelle se traduit généralement par une forte valeur absolue de la différence des deux amplitudes. L'interface générique présentée dans ce chapitre (Section 4.2.2) pourrait facilement être adaptée pour permettre ce type de combinaison. Un simple script pourrait également être imaginé pour spécifier la fonction de combinaison avec en outre la possibilité de combiner un nombre quelconque de volumes.

Au sujet de l'algorithme de visualisation de grands horizons en niveaux de détails, notre contribution consiste essentiellement en l'implantation d'une combinaison de techniques existantes. Bien que nous ne souhaitions pas lui faire occuper une position centrale dans la mesure où il ne s'agit que d'un élément annexe dans notre travail sur la visualisation volumique, il pourrait être intéressant de l'enrichir de techniques de compression [LOSASSO F. et HOPPE H., 2004] et/ou de mettre au point un système de cache pour le chargement sélectif des données en mémoire. Un tel système fera l'objet du chapitre suivant pour le cas particulier des données volumiques mais s'adapte parfaitement au format de stockage des horizons sismiques.

Finalement, nous voudrions insister en dernier lieu sur une tendance récente dans le domaine de la visualisation qui consiste à détourner le processeur graphique de son utilisation première qu'est la visualisation. En effet, le développement du matériel graphique évoqué à la Section 2.3.1 du Chapitre 2, tant du point de vue de sa flexibilité que de sa puissance, a conduit à de véritables architectures parallèles SIMD[15] [KILGARIFF E. et FERNANDO R., 2005] rivalisant avec les CPU dans la résolution de problèmes fortement parallèles. Un processeur NVIDIA Ge-

[15]de l'anglais "Simple Instruction Multiple Data"

Force 6 permet en effet d'atteindre plus de 50 GFlops[16] [Owens J.D., 2005], ce qui incite à en tirer partie pour des calculs habituellement exécutés sur le processeur central. Ce qui n'était au départ qu'un phénomène de mode a véritablement donné naissance à une branche à part entière de la visualisation scientifique portant le nom de GPGPU, de l'expression anglophone "General-Purpose Computation on Graphics Hardware" [GPGPU, –]. Une étude détaillée des travaux récents dans ce domaine est proposée dans [Owens J.D. *et al.*, 2005]. La Figure 4.20 donne un exemple très basique de calcul généraliste sur le GPU. Ainsi, l'ensemble de Mandelbrot est calculé par le processeur graphique en temps réel sur un terrain décimé par l'algorithme décrit en Section 4.3.3 de ce chapitre. Le calcul de l'équation caractérisant l'appartenance d'un point à l'ensemble est effectué par pixel à l'aide d'un programme de fragment. Cet exemple simple laisse entrevoir la possibilité dans un avenir proche de calculer en temps réel les attributs sismiques sur le processeur graphique au moment de les visualiser. La principale limitation actuelle des GPU est la précision de l'arithmétique flottante qu'ils supportent (simple précision, 32 bits), mais la tendance observée laisse espérer leur évolution vers le support de la double précision (64 bits), nécessaire au calcul scientifique. D'autre part, les détracteurs du GPGPU mettent également souvent en avant le coût de communication lors de l'envoi des données en mémoire graphique et plus encore lors du rapatriement du résultat en mémoire centrale. Dans notre cas, ce dernier transfert n'interviendrait pas dans la mesure où nous nous contenterions de visualiser le résultat sur une coupe horizon.

Nous avons évoqué à plusieurs reprises au cours de ce chapitre le problème de la taille parfois prohibitive des données volumiques impliquées dans l'interprétation sismique. Notre exposé des techniques de visualisation ne s'est cependant pas, jusqu'à présent, posé le problème des contraintes matérielles qu'imposait l'utilisation de stations d'interprétation. C'est de ce problème particulier que traite le chapitre suivant, en se plaçant dans le contexte de l'interprétation sismique avec couplage de la visualisation et des calculs sur une station d'interprétation standard.

[16] de l'anglais "giga floating point operations per second" (milliards d'opération en virgule flottante à la seconde)

Chapitre 5

Mémoire virtuelle pour la visualisation et les calculs

L'augmentation constante de la taille des volumes sismiques observée depuis une dizaine d'années a principalement deux conséquences. Du point de vue du géophysicien et/ou du géologue chargés de l'interprétation, la quantité d'information disponible prend des proportions démesurées rendant d'autant plus difficile la détection d'événements importants. Ceci conduit à mettre en place des démarches organisées telles que l'alternance entre macro- et micro-interprétation (Section 1.2.3) afin d'appréhender les différentes échelles de raisonnement de manière efficace. D'autre part, des techniques de visualisation adaptées au contexte d'interprétation, données sismiques seules (Chapitre 3) ou données multimodales (Chapitre 4), permettent d'exploiter l'information de manière optimale. Cependant, du point de vue de l'architecte d'applications d'interprétation sismique, les volumes de données à manipuler (couramment de l'ordre de 10 Go) dépassent largement la capacité de traitement temps réel des stations d'interprétation standards (typiquement plusieurs centaines de mégaoctets). Ce chapitre a pour objectif la prise en compte de ces limitations dans les algorithmes présentés précédemment. Après un exposé des contraintes liées au matériel disponible sur une station d'interprétation standard, notamment en termes de quantité de mémoire, un certain nombre de techniques destinées à la visualisation de volumes de grande taille pourront être présentées. Le contexte particulier de l'interprétation de données sismiques impliquant d'autre part un couplage de la visualisation et des calculs ne permet pas le recours à un certain nombre d'entre elles, ce qui le cas échéant sera justifié. Nous détaillerons alors le système mis en place, les implications de ce dernier sur le moteur de rendu volumique et terminerons en donnant un exemple d'utilisation sur un jeu de données de 20,8 Go au total, composé de quatre volumes d'information de 5,2 Go chacun. Ce chapitre sera avant tout l'occasion d'introduire un certain nombre de notions relatives à la gestion de la mémoire sur un système simple avec en ligne de mire un système beaucoup plus complexe qui fera l'objet du Chapitre 6.

5.1 Contraintes matérielles inhérentes aux données massives

Cette section a pour but de résumer les principales contraintes matérielles intervenant dans la manipulation de données volumiques massives dans un contexte de visualisation couplée aux calculs. La notion de quantité de mémoire disponible occupe une place particulièrement importante dans la mesure où, comme nous le verrons, le traitement en temps réel de données

par le processeur suppose leur présence en mémoire. Nous présenterons les différents types de mémoires disponibles sur une station de travail, d'abord de manière détaillée indépendamment pour la visualisation et les calculs, puis de manière synthétique dans un contexte de couplage des deux approches.

5.1.1 Hiérarchie de mémoire et notion de cache

Le traitement de données, à des fins de visualisation ou de calculs classiques, fait dans les deux cas intervenir un processeur. Comme nous l'avons vu à la Section 2.3.1, les traitements graphiques font l'objet d'un processeur spécialisé appelé GPU, par analogie avec CPU qui fait référence au processeur central, en charge de tous les calculs non graphiques. Chacun de ces deux processeurs dispose de son propre espace mémoire, lequel est organisé de manière hiérarchique par ordre croissant de temps d'accès aux données. L'expression *hiérarchie de mémoire* fait donc référence à cette organisation par niveaux d'accès successifs avec en terminaison le disque qui stocke l'information de manière durable sous forme de fichiers.

Processeur central et hiérarchie de mémoire

L'accroissement constant de la fréquence des processeurs au cours des dix dernières années a mis aujourd'hui à notre disposition des processeurs dont les fréquences dépassent les 3 GHz, ce qui permet, en théorie, l'exécution de plus de trois milliards d'opérations à la seconde. En pratique, certaines opérations prenant plusieurs cycles, ce chiffre est à relativiser. Le temps d'accès aux données limite également les performances dans des proportions très importantes. En prenant en compte ces deux facteurs, les performances réelles sont de quelques centaines de millions d'opérations par seconde (à comparer au maximum théorique de trois milliards). Malgré tout, Le coût de développement de mémoires capables de tenir le rythme imposé est particulièrement élevé. La solution à ce problème est la mise en place de hiérarchies de mémoire. Il s'agit en fait de disposer successivement plusieurs niveaux de mémoire, de temps d'accès décroissant mais de taille croissante. Dans une telle hiérarchie, chaque niveau joue le rôle de *cache* du niveau d'accès plus lent immédiatement placé derrière. La partie supérieure de la Figure 5.1 détaille la hiérarchie de mémoire du processeur central. Les caches L1 et L2, de petite taille, sont d'accès plus rapide que la *mémoire centrale*, couramment appelée *RAM* (de l'anglais "Random Access Memory") qui elle-même sert de cache du disque dur. D'une manière générale, le traitement en temps réel des données par le processeur central nécessite leur présence en RAM. En effet, le temps d'accès à la mémoire centrale est de l'ordre de la centaine de nanosecondes tandis que celui pour un disque standard est de l'ordre de la dizaine de millisecondes [Silberschatz A. *et al.*, 2005]. L'accès au cache L1 est inférieur à la nanoseconde. À échelle humaine, si nous ramenions, par l'imagination, le temps d'accès à la mémoire centrale de l'ordre de la minute (moins d'une seconde pour le cache L1), l'accès au disque est de l'ordre de 70 jours, ce qui fait environ deux mois et demi. Il est donc important pour une application qu'à chaque instant les données à traiter soient disponibles en RAM, ce qui nécessite la mise en place de techniques particulières lorsque celle-ci est présente en quantité insuffisante. Nous aurons l'occasion de revenir sur ces techniques en Section 5.3.

Processeur graphique et hiérarchie de mémoire

Un principe similaire de hiérarchie de mémoire existe dans le cas du processeur graphique. En effet, la mémoire RAM dispose de segments réservés par le système d'exploitation pour le

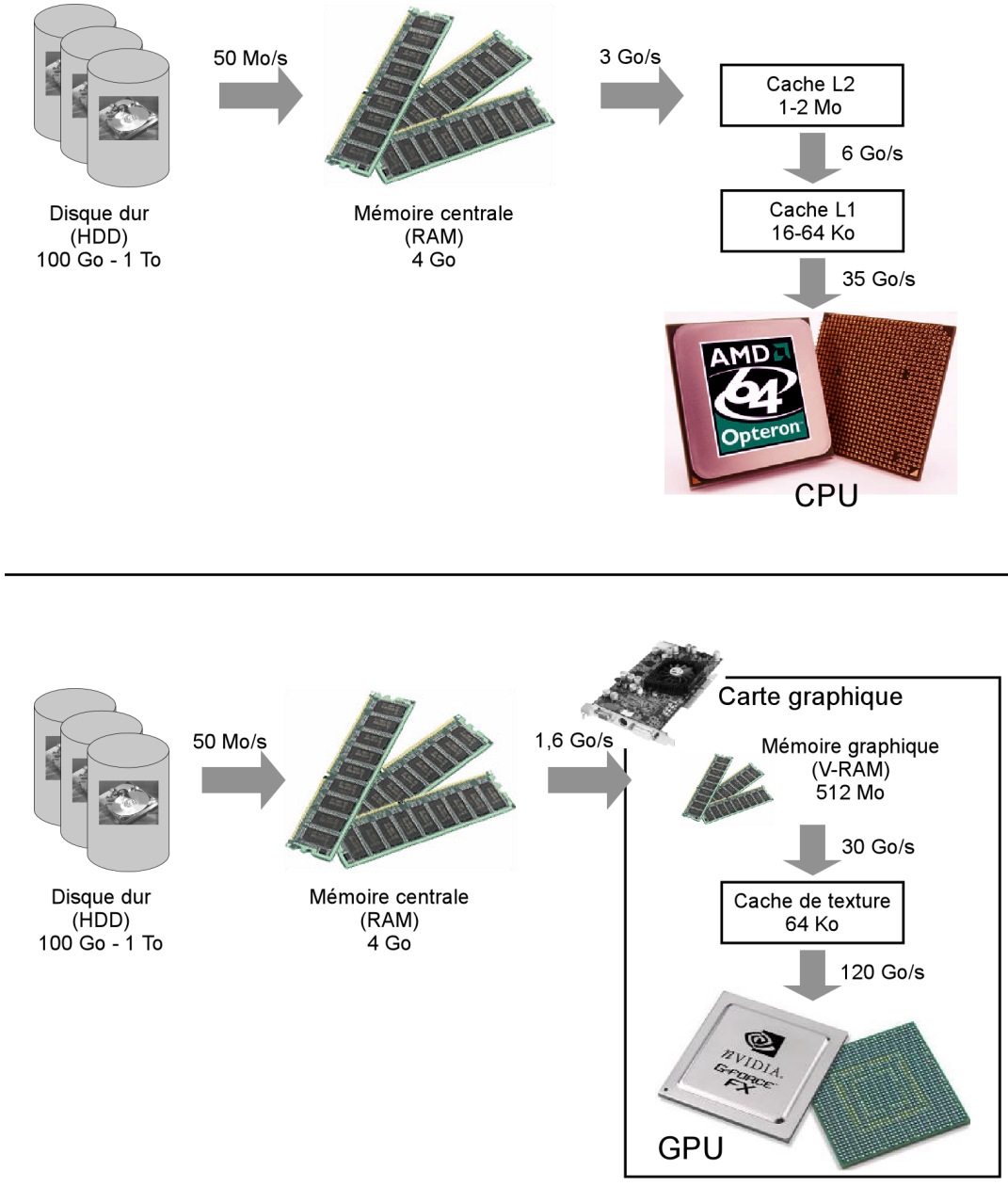

FIG. 5.1 – Hiérarchies de mémoire comparées du processeur central (haut) et du processeur graphique (bas). La notion de *cache* est généralisée, chaque niveau jouant le rôle de cache du niveau suivant. Les bandes passantes sont données à titre indicatif pour un accès unidirectionnel en lecture aux données (obtenues à partir de SiSoftware Sandra 2007 [SISOFTWARE SANDRA 2007, –]).

contrôleur graphique. Cette zone particulière porte le nom de *mémoire AGP* (de l'anglais "Accelerated Graphics Port"). Ainsi, comme le montre la partie inférieure de la Figure 5.1 (modifiée et réactualisée d'après [Rezk-Salama C. *et al.*, 2004]), les données graphiques (textures, ...) sont directement transférées vers la mémoire de la carte graphique. Cette dernière porte le nom de *V-RAM* pour *Video-RAM*, par analogie avec la RAM classique en mémoire centrale. Un niveau intermédiaire supplémentaire existe entre la V-RAM et le processeur graphique : le *cache de texture* dont la taille et la bande passante sont des données sensibles sur lesquelles les fabricants de cartes graphiques communiquent peu. Nous verrons cependant au Chapitre 6 qu'il est possible d'accéder expérimentalement à sa taille, ce qui nous a permis de l'évaluer à environ 64 Ko sur une carte NVIDIA GeForce 6800 Ultra. [Rezk-Salama C. *et al.*, 2004] proposent un parallèle entre les niveaux de cache du processeur central et la hiérarchie de mémoire du processeur graphique. Ainsi, ils assimilent le cache de texture au cache L1 du processeur central, la mémoire graphique V-RAM au cache L2 et la mémoire RAM à un niveau de cache L3. La notion de temps réel dans le cadre de la visualisation se définit en termes de taux de rafraîchissement limite de l'écran en hertz (Hz) ou, ce qui revient au même, en nombre limite d'images par secondes (fps), la limite étant communément de l'ordre de 20 fps. Or, le pic de transfert entre la mémoire RAM et la mémoire V-RAM, de l'ordre de 1,6 Go/s théorique, dépasse rarement en pratique 1 Go/s comme nous pourrons le voir au cours du Chapitre 6. Ceci signifie que le traitement à des fins de visualisation par le processeur graphique de 1 Go de données stockées en mémoire centrale RAM ne permet pas d'afficher plus d'une image par seconde. Ce calcul très approximatif ne tient pas compte du temps de traitement qui ne fait qu'aggraver la situation. Il est donc nécessaire, dans le cas de la visualisation, de disposer de l'information à visualiser dans les niveaux de cache supérieurs L1 et L2, c'est-à-dire au minimum en mémoire V-RAM dont la bande passante vers le cache de texture de 30 Go/s permettra d'atteindre approximativement les 30 fps.

5.1.2 Synthèse de la hiérarchie de mémoire pour la visualisation et les calculs

Dans le contexte de l'interprétation sismique, la visualisation des données n'est pas une finalité. En effet, la construction du modèle structural notamment repose sur des algorithmes de traitement exécutés par le processeur central (Section 4.3.1). De même, le calcul d'attributs sismiques ou de cartes volumiques de distance fait intervenir la manipulation de données volumiques par le processeur central (Section 4.1.1). Les contraintes de manipulation de données massives se trouvent donc intervenir à deux niveaux séparément : celui du processeur central et celui du processeur graphique. En tenant compte des conditions de temps réel qui ont été mises en évidence à la section précédente dans ces deux contextes particuliers, il est possible de proposer une hiérarchie de mémoire simplifiée qui fait la synthèse des contraintes matérielles dans un scénario de couplage de la visualisation et des calculs. Cette hiérarchie apparaît à la Figure 5.2 où les données volumiques sont stockées sur disque. La mémoire RAM joue le rôle de cache du disque pour les calculs classiques en temps réel sur le processeur central tandis que la mémoire V-RAM joue le rôle de cache de la RAM pour les calculs graphiques en temps réel sur le processeur graphique.

Lorsque le volume ne peut être entièrement chargé en V-RAM, ou même en RAM, il faut recourir à des techniques qui réduisent la taille des données nécessaires à un instant particulier. La section suivante présentera les différentes stratégies mises en œuvre dans la littérature pour le cas de la visualisation seule. La portée de chacune d'entre elles dans un contexte de couplage en temps réel de la visualisation et des calculs sur des données sismiques sera évaluée.

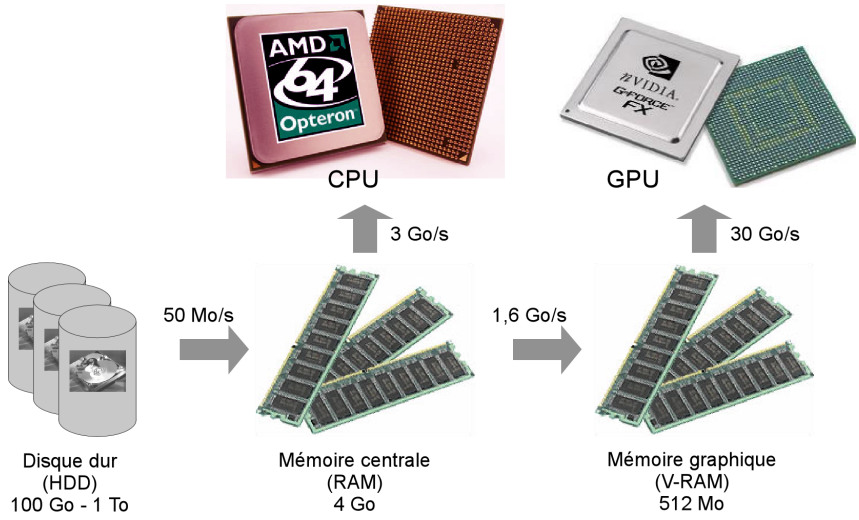

FIG. 5.2 – Hiérarchie de mémoire synthétique dans un scénario de couplage de la visualisation et des calculs. La mémoire V-RAM joue le rôle de cache de la mémoire RAM qui elle-même joue le rôle de cache du disque où les données sont stockées de manière durable.

5.2 Visualisation de volumes de grande taille : état de l'art

L'objectif de cette section est de dresser un état de l'art des techniques de visualisation de volumes de grande taille sur une station de travail équipée de matériel graphique standard (Section 2.3.1). Pour la plupart implantées dans le cadre de la visualisation de données médicales, certaines d'entre elles ne sont pas directement exploitables pour la visualisation de données sismiques. D'autre part, le cadre restrictif de la visualisation ne permet pas d'envisager dans tous les cas une extension au scénario plus large de couplage avec les calculs sur le processeur central. Il nous paraît cependant important de dresser cette liste de manière exhaustive.

5.2.1 Décomposition en briques et sondes volumiques

La première approche qui vient naturellement à l'esprit dans la gestion de volumes dont la taille dépasse celle de la mémoire graphique est une décomposition des données en sous-volumes élémentaires, également appelés *briques* [GRZESZCZUK R. *et al.*, 1998]. Ces briques élémentaires (Figure 5.3-A) peuvent être successivement chargées en mémoire graphique sous forme de textures 3D et rendues séparément dans un ordre défini, de l'arrière vers l'avant ou inversement, pour constituer l'image finale. Il est cependant important dans ce cas de tenir compte du principe de fonctionnement de l'interpolation trilinéaire accélérée par le matériel graphique qui sacrifie un demi-texel en bordure de texture. Plus précisément, il est important de noter que l'interpolation fait nécessairement intervenir dans les zones de bordure un texel virtuel transparent à l'extérieur de la texture. Peu gênant en bordure de volume, ce phénomène dégrade significativement la frontière des briques (Figure 5.3-B). Il est dans ce cas nécessaire de dupliquer les voxels situés dans la zone de contact entre deux briques adjacentes. Comme le montre la Figure 5.3-C, la duplication d'un seul voxel, normalement suffisante pour une interpolation trilinéaire correcte, s'avère insuffisante dans le cas de rendu volumique préintégré. En effet, la lecture de la valeur en sortie de tranche (Section 3.3.1) avec interpolation correcte nécessite au minimum la duplication de deux voxels (Figure 5.3-D).

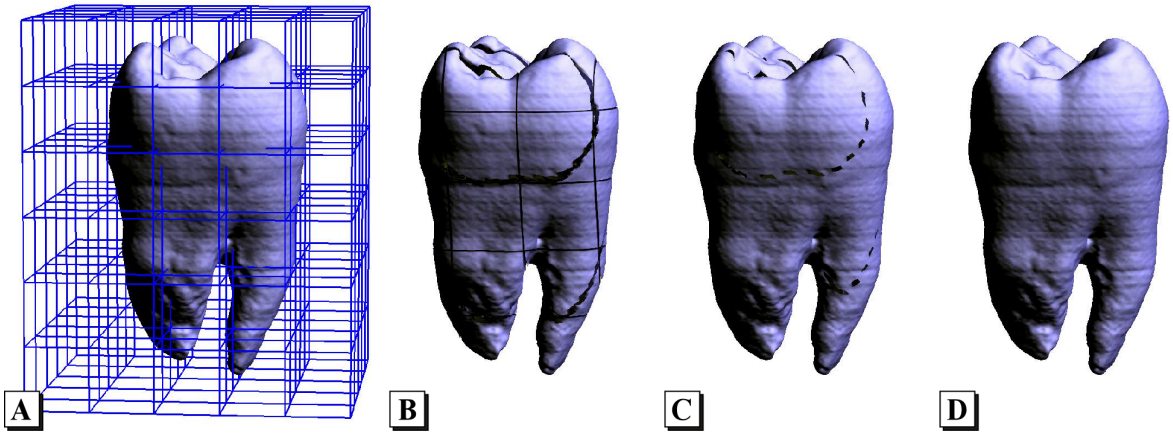

FIG. 5.3 – Duplication des voxels à la frontière entre deux briques pour une interpolation trilinéaire correcte dans le rendu volumique préintégré de données de scanner d'une dent. La décomposition en briques du volume (A) génère des artefacts aux frontières de briques (B) que la duplication d'un seul voxel ne suffit pas à faire disparaître (C). La duplication de deux voxels au minimum parvient à fournir un résultat correct (D).

À elle seule, cette décomposition ne résout pas le problème de la visualisation en temps réel de grandes quantités de données volumiques. En effet, elle permet tout au moins un rendu progressif de ces derniers par blocs chargeables séparément en mémoire graphique. Ceci suppose une consommation significative de la bande passante entre la mémoire AGP et la mémoire graphique V-RAM qui réduit considérablement les performances. Cependant, dans le cas de la visualisation en sismique, l'utilisation de sondes volumiques que l'utilisateur déplace dans le volume global permet, si toutefois cette sonde ne dépasse pas la taille de la mémoire graphique, de ne charger que les briques correspondantes [MARSH A.J., 2001]. Dans la littérature anglophone, cette technique porte le nom de *"volume roaming"* [GRZESZCZUK R. *et al.*, 1998]. De nombreux domaines d'application ont adopté une approche similaire par sonde volumique et des systèmes tels qu'OpenGL Volumizer [BHANIRAMKA P. et DEMANGE Y., 2002] ou encore Octreemizer [PLATE J. *et al.*, 2002] pour le cas des données sismiques sont des exemples de mise en œuvre de ce concept. Tous deux reposent sur des méthodes efficaces de gestion de la mémoire dont certaines sont au cœur du système qui sera présenté en Section 5.3.

5.2.2 Multirésolution

Les techniques de rendu par sondes volumiques dont la taille est limitée par la quantité de mémoire graphique détournent le problème en réduisant, à un instant donné, l'extension du domaine d'intérêt. Particulièrement bien adaptées à la démarche d'interprétation des données sismiques, de telles sondes peuvent apparaître insuffisantes dans certains cas, notamment lorsqu'il est question de données médicales dont le praticien veut une vue d'ensemble. Un moyen d'économiser la bande passante entre la mémoire AGP et la mémoire graphique dans ce cas est le recours à des niveaux de résolution variable. Ainsi, LAMAR *et al.* introduisent dans [LAMAR E. *et al.*, 1999] une décomposition du volume en *arbre octal*, plus généralement appelé *octree*, qui étend la notion d'arbre quaternaire pour la décomposition de grilles régulières 2D (Section 4.3.3) au cas de grilles régulières volumiques. Chaque nœud de l'arbre correspond à une brique de texture 3D. Les feuilles correspondent aux données initiales à résolution maximale tandis que les niveaux in-

termédiaires sont progressivement décimés par fusion de groupes de huit nœuds inférieurs. Huit voxels adjacents sont moyennés dans un seul voxel, ce qui assure une taille constante des textures à tous les niveaux de l'arbre. La construction d'un tel arbre peut se faire indifféremment de haut en bas ou de bas en haut. Par contre, le parcours en temps réel s'effectue toujours de haut en bas avec raffinement en fonction de divers critères comme des critères de visibilité faisant intervenir la distance au point de vue. Tout comme pour les niveaux de détail sur les surfaces (Section 4.3.3), la jonction entre deux niveaux de résolution différente est le lieu d'artefacts dont Weiler *et al.* s'affranchissent dans [Weiler M. *et al.*, 2000] en corrigeant l'opacité dans la fonction de transfert en fonction du degré de décimation.

Les systèmes de visualisation par sondes volumiques évoqués précédemment, OpenGL Volumizer [Bhaniramka P. et Demange Y., 2002] et Octreemizer [Plate J. *et al.*, 2002], font également intervenir la notion de résolutions multiples. L'affichage des niveaux de résolution grossiers intervient typiquement quand l'utilisateur déplace la sonde vers une zone encore inexplorée du volume dont les données ne sont pas chargées en mémoire graphique. Dans ce cas, des textures de résolution inférieure sont chargées de sorte que la consommation de bande passante ne dépasse pas un seuil critique, et qu'ainsi soit maintenu un taux de rafraîchissement de l'écran constant. Les textures de résolution maximale sont progressivement chargées lorsque l'utilisateur stoppe le mouvement de la sonde.

Il est cependant important de préciser que les fréquences de variation verticale des données sismiques sont extrêmement élevées contrairement aux données médicales dont les trois axes jouent au contraire le même rôle. De ce fait, le comportement de ces approches à résolutions multiples par fusion de voxels perd largement de son intérêt. En effet, il est généralement impossible d'observer la moindre signature d'un événement important avant d'avoir atteint le niveau de résolution maximale. L'objectif principal d'une telle approche étant d'assurer une manipulation interactive des paramètres de l'image par l'utilisateur (point de vue, sonde, fonction de transfert, ...), nous utilisons une méthode alternative répandue dans les applications de visualisation de données sismiques qui remplit ces objectifs. Nous appelons cette méthode le *rendu progressif interactif* et elle sera décrite en Section 5.4.3.

5.2.3 Compression

Pour réduire la consommation mémoire des volumes de données, une approche complémentaire des niveaux de résolution variable est l'utilisation de techniques de compression. La mise en œuvre la plus simple consiste à recourir au matériel graphique qui expose des options de compression de textures avec perte. Celles-ci sont accessibles à travers l'extension OpenGL `EXT_texture_compression_s3tc` pour les textures 2D, les concepts ayant été portés au cas des textures 3D dans l'extension propriétaire NVIDIA `NV_texture_compression_vtc`. Cependant, l'encodage travaillant sur des groupes de 4^3 texels, les résultats obtenus par cette voie souffrent d'effets de blocs sur des données fortement hétérogènes.

Une approche alternative utilise des *ondelettes* dont les applications en informatique graphique ont été largement explorées [Stollnitz E.J. *et al.*, 1995a, Stollnitz E.J. *et al.*, 1995b]. Dans [Nguyen K.G. et Saupe D., 2001], après décomposition du volume en briques, Nguyen et Saupe compressent les données de chaque brique individuellement à l'aide d'une projection sur une base d'ondelettes suivie d'un encodage des coefficients. Il s'agit là encore d'une compression avec perte que Güthe et Strasser combinent dans [Guthe S. et Strasser W., 2001] avec une approche multirésolution pour le rendu en temps réel de plusieurs gigaoctets de données 4D. Il est cependant important de souligner que, dans chacune de ces deux approches, la

compression est implantée sur le processeur central, ce qui signifie que les données doivent subir une décompression en mémoire AGP avant leur chargement en mémoire graphique V-RAM. L'intérêt pour les auteurs est de pouvoir charger l'intégralité des données en mémoire centrale, ce qui évite l'accès au disque. D'autre part, dans leur système de compression de données 4D avec multirésolution, Güthe et Strasser ne visualisent jamais l'ensemble des données dans la même image. L'autre limitation de l'utilisation d'ondelettes est que le taux de compression devient intéressant pour des données contenant des régions à faible variation et des zones de hautes fréquences localisées. Ces caractéristiques se retrouvent fréquemment dans les données médicales qui contiennent localement des limites d'organes, mais une fois de plus ne s'appliquent pas au cas des données sismiques où les hautes fréquences sont omniprésentes. À noter cependant une technique récente qui permet de résoudre partiellement le problème en remplaçant les ondelettes par leur équivalent orienté dans une direction arbitraire [Zarantonello S.E. and Bevc D., 2005]. En anglais, ces outils mathématiques portent le nom de *"ridgelets"* (ce qui pourrait se traduire en français par "crêtelettes"). L'approche semble prometteuse, cependant les auteurs ne fournissent aucun temps de compression. La complexité de leur algorithme d'encodage/décodage est en $O(n^2 \log(n))$ pour un cube de dimension n^3 avec des "ridgelets" 2D, ce qui est relativement important pour espérer un décodage en temps réel de grandes quantités de données. Par conséquent, leur technique est plutôt destinée au partage de données sismiques à travers Internet où les contraintes de temps réel ne sont pas les mêmes que dans notre cas.

Pour réduire la consommation de mémoire graphique, certains auteurs proposent une forme de compression originale par empaquetage de blocs de données homogènes. La taille en mémoire des régions homogènes du volume est ainsi considérablement réduite, ce qui sur certains modèles volumiques de données médicales réduit de moitié la taille totale des textures. Le décodage des blocs est exécuté en temps réel sur le processeur graphique, dans certains cas au niveau du processeur de fragment [Kraus M. et Ertl T., 2002], dans d'autres au niveau du processeur de sommet [Li W. et Kaufman A., 2003, Li W. *et al.*, 2003]. Cependant, là encore l'hétérogénéité des données sismiques ne permet pas d'exploiter pleinement ces techniques.

Finalement, le dernier type de compression repose sur un *encodage de vecteurs* dont l'implantation du décodage sur le processeur de fragment permet un rendu directement depuis l'espace de compression [Schneider J. et Westermann R., 2003]. Le principe de l'encodage de vecteurs consiste à stocker des vecteurs de valeurs à l'aide d'un simple indice qui sert d'entrée dans une table. L'ensemble des vecteurs de la table et des indices de chacun des vecteurs initiaux est une représentation beaucoup plus compacte des données. Les vecteurs stockés dans la table sont obtenus par une *analyse en composantes principales* (*ACP*) qui assure une fidélité maximale dans la restitution des vecteurs initiaux. Les données volumiques sont en fait compressées en deux étapes. Tout d'abord, des blocs 4^3 sont réduits en blocs 2^3 par fusion de voxels. Le vecteur de différence entre les deux blocs contenant 64 composantes est encodé comme décrit plus haut. Les blocs 2^3 sont à nouveau réduits à une seule valeur, et le vecteur de différence contenant 8 composantes est à son tour encodé. Ainsi, chaque bloc 4^3 est représenté de manière compacte par une valeur scalaire et deux indices pour la lecture des vecteurs de différence, ce qui permet de les stocker dans des textures 3D définies dans un espace *RGB* à trois composantes. Les tables d'accès aux vecteurs de différence sont quant à elles stockées sous forme de textures 2D en mémoire graphique également. Le décodage en temps réel de ces blocs nécessite cependant un certain nombre d'accès de texture successifs dits de lecture dépendante, c'est-à-dire dont l'entrée provient de la sortie d'une autre texture. Ceci, couplé à l'utilisation de textures différentes provoque de nombreux transferts de données entre la mémoire graphique V-RAM et le cache de texture réduisant ainsi significativement les performances. Si d'autre part nous

tenons compte du caractère extrêmement spécialisé d'une telle approche clairement orientée vers la visualisation pure, elle n'apparaît pas comme un candidat idéal pour le scénario de couplage de la visualisation et des calculs dans lequel nous nous trouvons.

De toutes ces techniques de visualisation de volumes de données de grande taille, l'approche par décomposition en briques associée à un système de rendu par sonde volumique est la plus en adéquation avec les attentes des utilisateurs d'applications d'interprétation sismique. D'autre part, un système de sondes est déjà en place dans le module VolumeExplorer du logiciel Gocad (Section 1.3). La section suivante abordera donc le mécanisme de gestion de la mémoire mis en œuvre pour permettre de coupler efficacement la visualisation de sondes et les calculs sur des volumes de données sismiques de grande taille.

5.3 Un système de cache hiérarchique couplant visualisation et calculs

Cette section a pour objectif de donner les détails du système de gestion de la mémoire que nous avons implanté dans le logiciel Gocad pour permettre une manipulation en temps réel de données volumiques de grande taille à des fins de calculs et/ou de visualisation. Ce système prend en compte les contraintes matérielles liées à la quantité de mémoire limitée dont disposent les stations d'interprétation (Section 5.1). De manière schématique, il est possible de fournir une représentation hiérarchique de ces contraintes (Figure 5.2), hiérarchie que nous retrouverons naturellement dans la conception du système. D'autre part, ces contraintes sont parfaitement connues des concepteurs de systèmes d'exploitation dont la complexité dépasse largement celle d'une application d'interprétation sismique. La solution mise en place par ces derniers porte de nom de *mémoire virtuelle*, dont nous nous sommes largement inspirés. Il est donc important d'en développer les grands principes. Pour plus de détails, le lecteur pourra se reporter à [Silberschatz A. *et al.*, 2005] ou [Tanenbaum A.S., 2001].

5.3.1 Pagination et mémoire virtuelle

La quantité de mémoire physique disponible sur une station de travail est un facteur limitant pour l'exécution de certaines applications dont l'espace mémoire requis dépasse la limite disponible. Pour faire face à cette situation, les systèmes d'exploitation ont recours à la notion de *mémoire virtuelle* qui consiste à simuler de manière transparente pour l'application la présence d'une grande quantité de mémoire. Ceci suppose un découplage entre la mémoire physique et l'espace mémoire continu manipulé par les applications. Une interface appelée *unité de traduction d'adresses* ou *MMU* (de l'anglais "Memory Management Unit") assure ce découplage. L'application manipule des adresses abstraites, dites virtuelles, indépendantes de leur localisation réelle définie par l'adresse physique. L'espace de mémoire virtuel étant plus grand que l'espace physique, le surplus est stocké dans un niveau inférieur de la hiérarchie (le disque dans ce cas) et chargé si nécessaire. Il est important de bien distinguer la notion de cache de celle de mémoire virtuelle qui est une implantation logicielle permettant de tirer partie de la hiérarchie de mémoire qu'exposent les caches successifs. Par extension, l'expression *système de cache* ou tout simplement cache est employée pour faire allusion à l'ensemble formé par la hiérarchie de caches et la mémoire virtuelle associée. À noter que la gestion des caches L1 et L2 du processeur

central (Figure 5.1), hautement sensible, est implantée au niveau des composants matériels du système.

En pratique, l'espace mémoire virtuel est découpé en blocs de taille constante appelés *pages* et gérés indépendamment les uns des autres. L'expression *pagination à la demande* (*"demand paging"* en anglais) traduit la mise à disposition en mémoire de pages référencées successivement par l'application. Une page présente en mémoire à un instant donné est dite *résidante*. Dans le cas contraire, une référence sur celle-ci génère un *défaut de page/cache* qui provoque un chargement depuis le disque. L'ensemble des pages résidantes est dit *ensemble résidant* (*"resident set"* en anglais). Le caractère résidant ou non d'une page dépend de nombreux facteurs dont la fréquence à laquelle l'application la référence. L'objectif d'un système de cache est de réduire au maximum les défauts de page pour réduire le temps moyen d'accès aux données. Ceci repose sur les propriétés de localité spatio-temporelle des programmes, à savoir qu'une référence à une donnée à un instant précis provoquera probablement une référence à une donnée voisine dans un avenir proche. L'algorithme dit de *"global clock"*[17] se fonde sur cette propriété pour décharger automatiquement les pages les plus anciennement référencées, et donc les moins susceptibles d'être accédées dans un avenir proche. Il repose sur l'utilisation de deux "aiguilles" circulant avec un décalage dans le temps sur les pages chargées en mémoire. Toute page n'ayant pas été référencée dans le délai écoulé entre le passage de chacune des deux "aiguilles" est déchargée.

Malheureusement, le recours au système de cache entre le disque et la mémoire centrale implanté dans les systèmes d'exploitation pour la visualisation de données volumiques massives se solde rapidement par une chute significative des performances [Cox M. et Ellsworth D., 1997]. Celle-ci est le résultat d'une augmentation soudaine du nombre de défauts de cache, phénomène portant le nom de *"thrashing"* [Silberschatz A. *et al.*, 2005]. Dans ce cas, il est nécessaire de mettre en œuvre une implantation de mémoire virtuelle propre à l'application. Les premiers systèmes apparus dans les années 1990 étaient destinés à la visualisation de champs d'écoulement dans des grilles régulières [Lane D., 1994, Cox M. et Ellsworth D., 1997] ou irrégulières [Ueng S.K. *et al.*, 1997]. Plus récemment, ils ont également servi à la visualisation de modèles polygonaux de très grande taille [Wald I. *et al.*, 2004].

Notre implantation de mémoire virtuelle repose sur une décomposition en briques comme celles vues en section précédente [Bhaniramka P. et Demange Y., 2002, Plate J. *et al.*, 2002]. Cependant, ces systèmes de mémoire virtuelle sont exclusivement dédiés à la visualisation. La mise à profit d'un cache de visualisation pour les calculs sur le processeur central suppose une hiérarchie avec plusieurs niveaux d'accès suivant la zone de mémoire concernée. Dans ce cas précis, il s'agit de fournir une entrée en mémoire RAM pour le processeur central et une entrée en mémoire V-RAM pour le processeur graphique.

5.3.2 Hiérarchie à deux niveaux d'accès

L'objectif du système de cache mis en place pour le couplage de la visualisation et des calculs est d'opérer une gestion transparente de la mémoire centrale et de la mémoire graphique disponibles. En ce qui concerne la mémoire centrale, nous avons introduit un niveau d'abstraction au sein de l'application qui permet de transformer un stockage classique de volume "à plat" par un stockage sous forme de briques avec gestion transparente des transferts de pages entre disque et mémoire. Ce mécanisme expose à l'application un espace de mémoire virtuel dans lequel l'ensemble du volume de données peut être adressé. Une adresse virtuelle est constituée d'un

[17] voir [Tanenbaum A.S., 2001] pour plus de détails

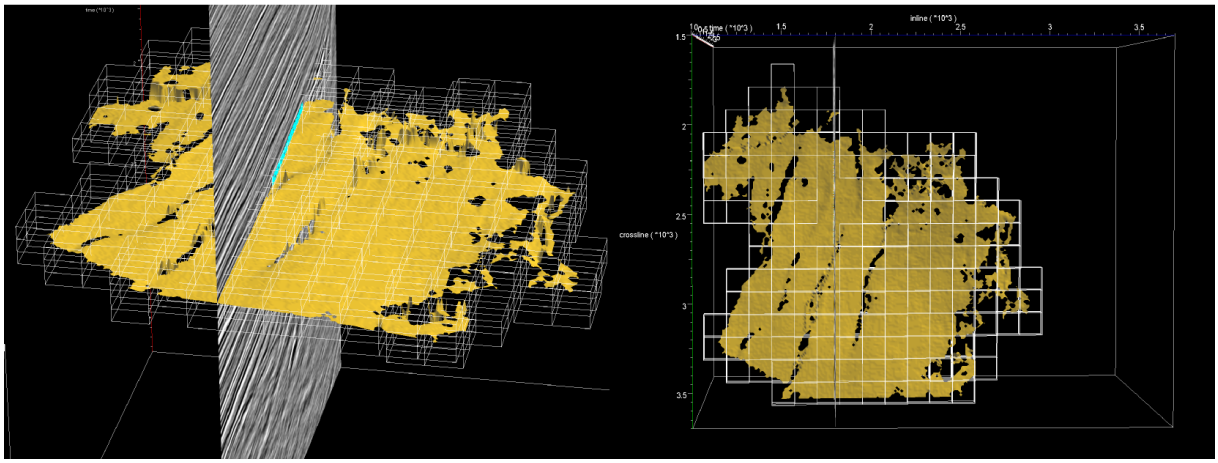

FIG. 5.4 – État de l'ensemble résidant (briques blanches) initialement vide après extraction semi-automatique d'un horizon sismique.

triplet d'indices $\{u, v, w\}$ qui définit la position du voxel dans le volume global. Une unité de traduction d'adresses se charge du calcul de la brique correspondant à l'adresse virtuelle d'une requête et lance son chargement si nécessaire. Après chargement, l'adresse physique de la requête est retournée. L'algorithme de gestion de requête a été implanté comme suit :

Traduction_adresse
Entrée
a_v : Adresse virtuelle dans le volume (triplet d'indices $\{u, v, w\}$)
Sortie
a_p : Adresse physique en mémoire
Début
$B \leftarrow$ Brique correspondant à a_v
Si B est chargée // *B est résidante*
$a_p \leftarrow$ Adresse physique de B + position de a_v dans B
Sinon // *Défaut de cache*
$a_p \leftarrow$ Adresse physique d'une brique chargée // *Section 5.3.3 pour plus de détails*
Charger B à l'adresse a_p
$a_p \leftarrow a_p$ + position de a_v dans B
Fin // *Si*
Fin

Lorsqu'une requête est émise, la brique correspondant à son adresse est intégralement chargée en mémoire. L'intérêt d'une telle décomposition en briques d'extension homogène dans toutes les directions de l'espace est de pouvoir exploiter la localité spatiale des algorithmes comme celui d'extraction d'horizon présenté en Section 4.3.1. La Figure 5.4 montre l'état d'un ensemble résidant initialement vide après extraction semi-automatique d'un horizon.

Les algorithmes de visualisation accèdent pour leur part aux données en mémoire graphique sous forme de textures. Ainsi, superposé à ce niveau de cache en mémoire centrale, un niveau d'abstraction de la texture implante un cache en mémoire graphique. Dans ce cas, l'adresse virtuelle est constituée d'un indice de brique à dessiner et l'adresse physique d'un indice de texture fourni par la librairie graphique OpenGL après chargement. Lors d'un défaut de cache en mémoire graphique, la requête est propagée au niveau de cache inférieur en mémoire cen-

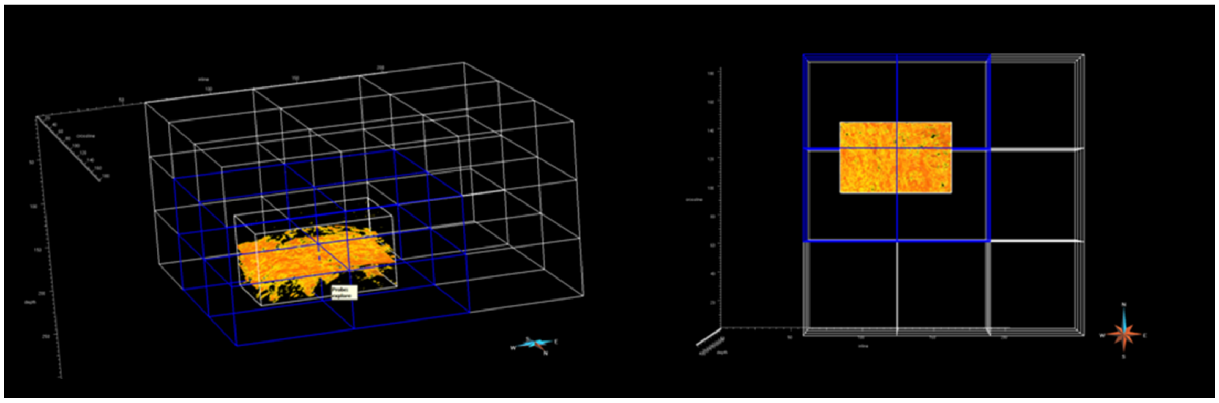

FIG. 5.5 – État des ensembles résidants des deux niveaux du cache hiérarchique lors de la visualisation d'une sonde volumique. Le chargement dans l'ensemble résidant en mémoire graphique (briques bleues) suppose la présence dans l'ensemble résidant en mémoire centrale (briques blanches).

trale. Si la brique n'est pas résidante, elle est chargée successivement en mémoire centrale puis en mémoire graphique. Ce type de cache à plusieurs niveaux est appelé *cache hiérarchique* [CHANKHUNTHOD A. *et al.*, 1996]. Comme le montre la Figure 5.5, dans un tel cache le chargement du niveau supérieur (mémoire graphique) suppose un passage dans les niveaux inférieurs (mémoire centrale).

Les différences entre ce système de décomposition en briques et ceux évoqués précédemment [BHANIRAMKA P. et DEMANGE Y., 2002, PLATE J. *et al.*, 2002] sont de plusieurs ordres. Tout d'abord, il expose une entrée en mémoire centrale avec un accès par voxel, ce que ne permettent pas des systèmes destinés à la visualisation seule. En effet, la granularité de leurs requêtes est au minimum de l'ordre d'une brique et l'adresse physique de retour est indifféremment un indice de texture en mémoire graphique comme pour le niveau supérieur de notre hiérarchie. D'autre part, les valeurs d'amplitude sismique sont généralement des valeurs à virgule flottante codées sur 32 bits. Or, les valeurs chargées en mémoire graphique sont généralement des entiers codés sur 8 bits afin d'économiser cette dernière. Les systèmes classiques se contentent de convertir les données sur disque et de ne manipuler que des valeurs entières. Cependant, cette conversion s'apparente à une compression avec perte que la plupart des algorithmes de traitement des données sismiques supportent mal. Nous conservons donc le format de stockage initial en mémoire centrale et insérons un filtre de conversion entre les deux niveaux du cache qui est appliqué lors du chargement en mémoire graphique. Finalement, ce système de gestion de la mémoire intervient pour toutes les données volumiques que l'application est amenée à manipuler en mémoire graphique comme en mémoire centrale, y compris lors du calcul de cubes d'attributs sismiques ou de cartes volumiques de distances. Il est donc nécessaire, à la différence des systèmes classiques, de supporter un **accès en écriture** au niveau des pages du cache que les systèmes de visualisation pure ne supportent pas. Pour cela, nous insérons un drapeau d'accès en écriture par page qui indique la nécessité ou non de mettre à jour les données sur disque lors de la libération de l'espace mémoire d'une page.

La libération de mémoire est en effet un élément important de tout système de cache. Dans l'algorithme présenté plus haut, le chargement d'une nouvelle page nécessite la mise à disposition d'une adresse physique valide. Pour cela, il est nécessaire de libérer l'espace mémoire occupé par une page précédemment chargée. Le choix de la page évincée repose sur une *stratégie de remplacement de page*.

5.3.3 Stratégie de remplacement de page

Par définition, la taille d'une mémoire virtuelle dépasse celle de la mémoire physique sous-jacente. À tout instant, l'ensemble résidant est donc un sous-ensemble de toutes les pages manipulées par l'application et possède une taille limite. Lors du chargement d'une nouvelle page, il est donc nécessaire de libérer une place dans cet ensemble. Le critère de sélection a une incidence significative sur l'évolution de l'état de l'ensemble résidant, sur la fréquence de défauts de pages et donc sur la performance du cache.

FIFO

La stratégie *FIFO*, de l'anglais "First-In, First-Out" (premier entré, premier sorti), consiste à évincer la page la plus ancienne de l'ensemble résidant. Ceci revient à ordonnancer les pages suivant l'**ordre de chargement**. Elle est mise en œuvre de manière extrêmement simple à l'aide d'une liste chaînée. Cependant, elle expose parfois le système à une anomalie connue sous le nom d'*anomalie de Belady*[18] qui provoque une recrudescence de défauts de cache lorsque la taille de l'ensemble résidant augmente. Ce résultat pour le moins inattendu conduit à utiliser des stratégies plus performantes.

LRU

La stratégie *LRU*, de l'anglais "Least Recently Used" (moins récemment utilisé), consiste à évincer la page qui a passé le plus de temps sans être référencée, ce qui revient à ordonnancer les pages suivant l'**ordre d'accès**. Dans ce cas, il est démontré que le comportement ne peut mettre en évidence d'anomalie de Belady [Silberschatz A. *et al.*, 2005]. Nous nous sommes donc orientés vers ce choix. D'un point de vue pratique, une stratégie LRU peut être implantée à l'aide soit d'une liste chaînée, soit d'un compteur de temps. Nous utilisons un compteur de temps relatif incrémenté à chaque référencement de page. Dans ce cas, il est important de tenir compte des éventuels dépassements de capacité. Nous utilisons un entier codé sur 32 bits. En théorie, il nous est donc possible d'effectuer 10^9 référencements de page par seconde pendant 585 ans avant de rencontrer un dépassement de capacité !

Nous avons précédemment évoqué l'implantation matérielle de la gestion des caches L1 et L2 du processeur central. Il en est de même pour le cache de texture de la mémoire graphique. De plus, la librairie graphique OpenGL expose une fonction `glPrioritizeTextures` qui permet de définir des ordres de priorité pour le chargement des textures dans le cache L1 du processeur graphique. Nous y avons donc recours lors de chaque référencement à une brique de texture. En effet, tout référencement de page suppose un accès imminent aux données correspondantes, d'où l'intérêt de s'assurer du chargement de la page dans le cache L1.

5.3.4 Taille de page

Le dernier point important dans l'implantation d'un système de cache est la taille de page utilisée. En effet, comme le montre la Figure 5.6, celle-ci a une incidence directe sur la fréquence de défauts de cache et donc sur les performances de l'application. Dans cet exemple d'extraction semi-automatique d'un horizon sismique, il apparaît que la page optimale est une brique de 64^3

[18] voir [Silberschatz A. *et al.*, 2005] pour plus de détails

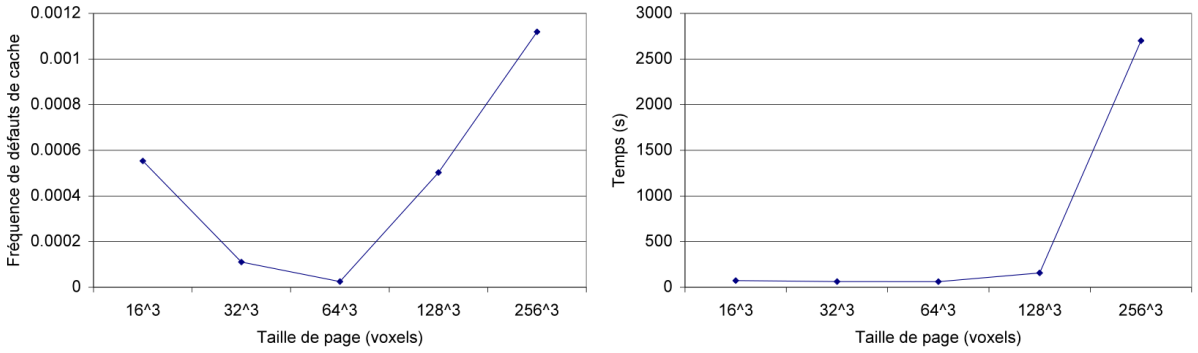

FIG. 5.6 – Influence de la taille de page sur l'efficacité du cache dans l'extraction semi-automatique d'horizons sismiques sur un cube de 5,2 Go (2608×661×811, 32 bits par voxel). La taille du cache est 320 Mo et les briques sont cubiques. L'horizon extrait couvre l'ensemble du cube à une résolution de 661×811. La courbe de gauche représente la fréquence de défauts de cache, tandis que la courbe de droite donne les temps d'extraction en secondes.

voxels. Il s'agit là bien évidemment d'un résultat empirique qui ne peut se vérifier sur l'ensemble des algorithmes de traitement en sismique. La forme générale des courbes est cependant riche en enseignements. En effet, de part et d'autre de cet optimum une taille inférieure et plus encore une taille supérieure provoquent une chute des performances. La réduction de taille génère en fait un surplus de travail dans la gestion des pages du cache, mais également réduit la localité spatiale des données. Cette réduction de localité spatiale explique également en partie le comportement des courbes lorsque la taille augmente. Dans ce cas, un facteur plus important encore est le surcoût des accès en lecture/écriture au disque lors d'un remplacement de page, une taille de 256^3 impliquant 64 fois plus de travail qu'une taille de 64^3.

L'utilisation de ce système de décomposition du volume en briques a un certain nombre d'implications sur le moteur de rendu volumique. La nécessité de dupliquer les voxels en bordure de brique en fait notamment partie (Section 5.2.1). Nous allons donc évoquer maintenant les ajustements les plus significatifs que nous avons apportés.

5.4 Ajustements du moteur de rendu volumique

Comme il a été vu aux Chapitres 3 et 4, les techniques de visualisation volumique implantées dans le cadre de ce mémoire reposent sur un moteur de rendu volumique par discrétisation implicite à base de textures 3D (Section 2.3.2). Dans le cas d'une décomposition du volume en briques, chaque brique de texture 3D est échantillonnée séparément à l'aide de tranches alignées parallèlement à l'écran. Du fait des propriétés d'associativité et de non commutativité des opérations de composition (Section 2.2.2), il est possible de faire un rendu séparé tant que l'ordre final de composition des briques est respecté. En marge de cela, nous avons intégré un certain nombre d'optimisations pour assurer un temps de rendu optimal.

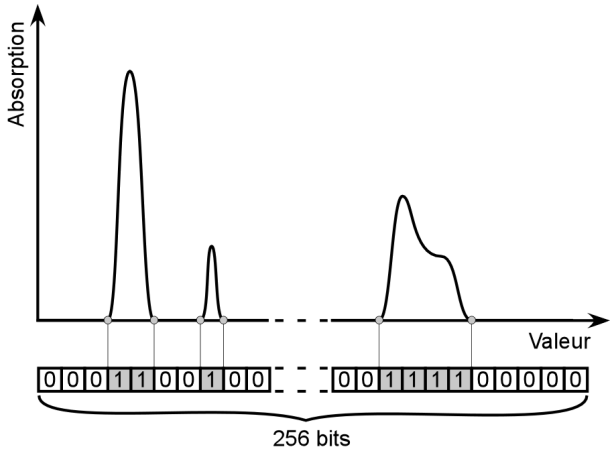

FIG. 5.7 – Principe d'encodage de la fonction de transfert dans un *histogramme binaire*. Un vecteur de 256 bits permet de stocker l'état binaire (entièrement transparente ou partiellement opaque) de chaque entrée dans la fonction de transfert.

5.4.1 Tranchage incrémental optimisé

Dans les techniques de décomposition du volume, une taille de brique trop faible peut avoir une influence néfaste sur le temps de rendu. En effet, bien que généralement limité par le processeur de fragment, le rendu volumique dans ce cas se retrouve limité par le processeur central chargé de calculer les polygones d'échantillonnage des briques. La méthode classique consiste à calculer séparément pour chaque tranche les points d'intersection avec les arêtes de la brique afin de reconstituer les polygones un à un. De même, il est nécessaire de calculer les coordonnées de texture 3D de chaque sommet. Pourtant, l'espacement entre les tranches étant constant, il est possible, à partir du premier polygone, de calculer les suivants par incréments successifs sur les sommets le long des arêtes intersectées. Cette technique porte le nom de *tranchage incrémental optimisé* [LI W. et KAUFMAN A., 2003] et nous permet de réduire significativement la charge du processeur central. LI et KAUFMAN utilisent la même méthode pour les coordonnées de texture cependant nous préférons dans ce cas recourir aux fonctionnalités de génération automatique qu'expose la librairie OpenGL. Les trois coordonnées sont calculées en fonction de la distance géométrique du point à trois plans spécifiés pour chaque brique.

5.4.2 Élimination des espaces vides

Dans certains cas, la fonction de transfert du rendu volumique conserve peu de parties opaques et le volume de données visualisé apparaît en grande partie transparent. C'est le cas par exemple lorsque l'interprétation porte sur les cœurs lenticulaires Z1 (Section 3.1.2) ou sur une surface d'isovaleur non polygonale d'une carte volumique de distance (Section 4.2.3). Sans traitement particulier, les zones transparentes sont pourtant affichées intégralement au même titre que les zones opaques, chargeant inutilement des données en mémoire graphique et gaspillant la puissance de traitement du processeur graphique. Dans un contexte de décomposition en briques, il est possible de prédétecter les briques entièrement transparentes afin de les exclure du rendu. Pour cela, nous utilisions la technique des *histogrammes binaires* (*"value histogram"* en anglais) [GAO J. *et al.*, 2005]. Les valeurs du champ scalaire en visualisation, tout comme les entrées dans la fonction de transfert, sont généralement codées sur des entiers 8-bit, ce qui laisse 256 valeurs possibles. Comme le montre la Figure 5.7, à chaque entrée dans la fonction

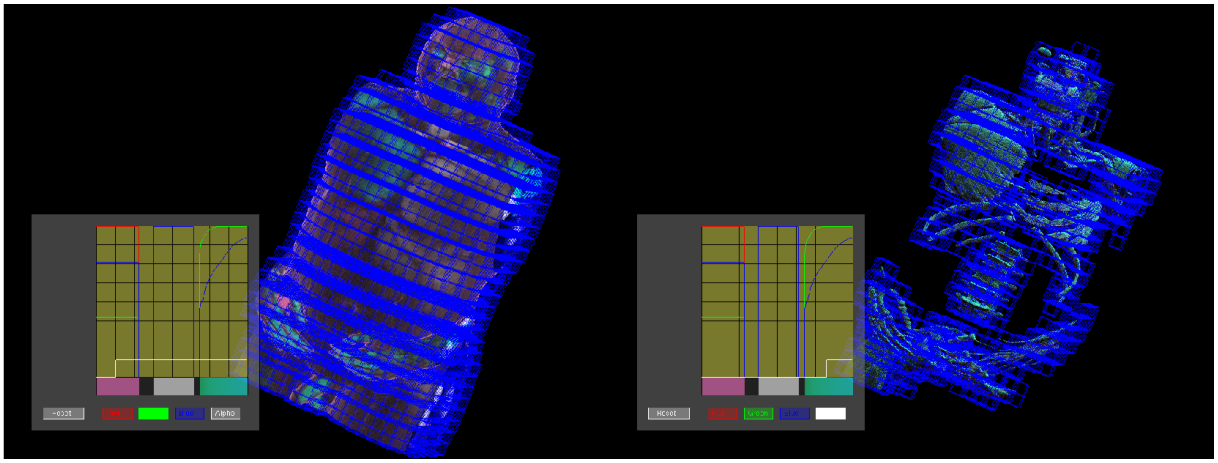

FIG. 5.8 – Élimination des espace vides sur le rendu volumique de données de scanner d'un être humain. Les briques non transparentes (bleues) sont déterminées en temps réel à l'aide d'opérations bit-à-bit sur des *histogrammes binaires* en fonction de la courbe d'opacité de la fonction de transfert (courbe blanche).

de transfert est associé un état binaire, entièrement transparente ou partiellement opaque, qui est codé sur un bit. L'ensemble de la fonction de transfert peut ainsi être encodé sur un vecteur de 256 bits. De même, l'histogramme des valeurs scalaires présentes dans une brique est encodé sur un vecteur similaire, l'état binaire étant défini par la présence ou non de l'entrée dans la brique. Le résultat de l'opération bit-à-bit AND entre ces deux histogrammes détermine en temps réel si une brique contient ou non une valeur opaque. La Figure 5.8 illustre l'utilisation de cette méthode pour l'exclusion des briques transparentes dans le rendu volumique de données de scanner d'un être humain. Les opérations bit-à-bit permettent de disqualifier en temps réel les briques rendues transparentes par l'édition de la courbe d'opacité de la fonction de transfert. Seul l'histogramme binaire de cette dernière est recalculé, celui des données visualisées restant inchangé.

5.4.3 Rendu progressif interactif

L'intérêt d'utiliser des résolutions multiples dans les systèmes de décomposition du volume en briques [BHANIRAMKA P. et DEMANGE Y., 2002, PLATE J. *et al.*, 2002] est d'assurer le maintien d'une valeur seuil du taux de rafraîchissement lorsque l'utilisateur manipule la sonde. À l'arrêt, les textures de résolution maximale sont progressivement chargées puis échantillonnées. Comme nous l'avons vu pour le cas de la sismique, ces approches ne permettent pas de produire une image informative tant que la résolution maximale n'est pas atteinte. Nous avons donc mis en œuvre une méthode alternative bien adaptée aux besoins des utilisateurs d'outils de rendu volumique en sismique. Elle consiste simplement à effectuer le rendu dans la *mémoire écran frontale* (*"front buffer"* en anglais), ce qui permet de voir apparaître les plans progressifs de l'arrière vers l'avant et ainsi se constituer l'image. Ceci permet de révéler les structures internes. Pour assurer l'interactivité, l'utilisateur peut reprendre la main après le rendu de chaque brique. Le moteur de rendu volumique est alors suspendu et seules les arêtes de la sonde sont dessinées pendant l'interaction avec l'utilisateur. Puis, à l'arrêt le moteur de rendu volumique est relancé. De cette manière, le système reste constamment réactif et l'image ne se fige jamais.

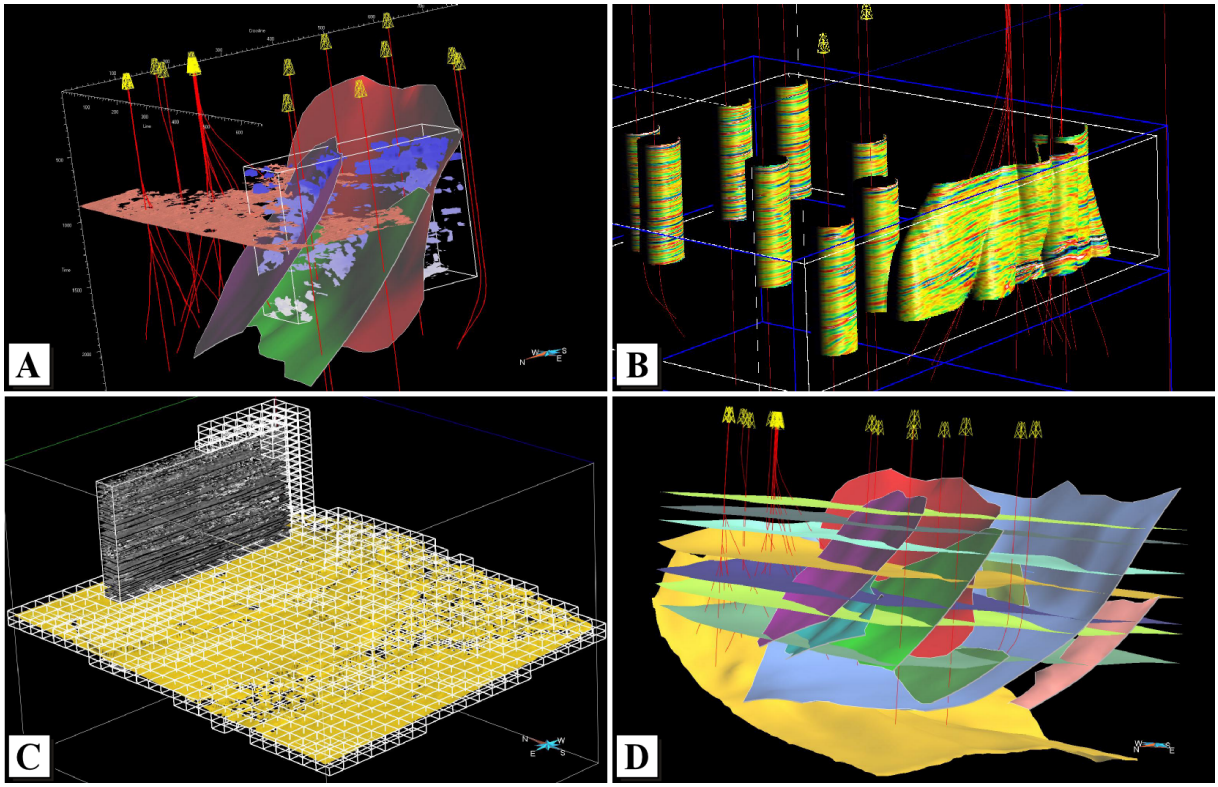

FIG. 5.9 – Exemples d'application du système de cache pour visualiser des sondes volumiques (A) et (B) ou coupler la visualisation et les calculs d'extraction (C) lors de la construction d'un modèle structural (D). (A) Rendu volumique préintégré de faibles valeurs de semblance qui font apparaître des discontinuités liées à la présence de failles. (B) Amplitudes sismiques plaquées sur une surface d'isovaleur non polygonale d'une carte de distance à un ensemble de puits de production. (C) L'ensemble résidant en mémoire centrale après extraction est représenté par les briques blanches. Le résidu de briques sur la sonde volumique correspond au chargement des textures dans le niveau supérieur du cache en mémoire graphique avant l'extraction de l'horizon.

Le système qui vient d'être décrit permet de coupler la visualisation et les calculs sur de grandes quantités de données volumiques. Il a été conçu avec pour objectif principal l'application à l'interprétation des données sismiques. Il est maintenant important de décrire en quelques lignes et quelques images des situations concrètes tirant profit de cette implantation dans le cadre applicatif fixé.

5.5 Cas d'étude : construction d'un modèle structural

Cette section concise a pour objectif de donner quelques exemples de situations concrètes d'utilisation du système transparent de gestion de la mémoire décrit au cours des deux sections précédentes. Il s'agit de démontrer à travers quatre images clés la façon dont il s'intègre dans les techniques de visualisation présentées tout au long de la Partie II sur la station d'interprétation. En toile de fond, nous considérerons un scénario de construction d'un modèle structural. Le cube sismique correspondant est un volume de 5,2 Go (2608×661×811, 32 bits par voxel). Quatre volumes d'information sont disponibles au total : les valeurs d'amplitude sismique initiales, un cube de semblance, un cube de vitesses sismiques et une carte volumique de distance à un

ensemble de puits de production. La carte de distance a en effet été calculée dans tout le volume et stockée sous la forme d'un nouvel attribut. Le volume total de données représente donc 20,8 Go. La station de travail utilisée pour cette étude est une bi Xeon 3,4 GHz avec 3 Go de RAM. La carte graphique est une Quadro FX 3400 disposant de 256 Mo de mémoire graphique et un bus PCI Express. La taille du niveau inférieur du cache en mémoire centrale est de 1 Go (256 Mo par volume de données) et le niveau supérieur utilise l'ensemble de la mémoire graphique disponible.

La Figure 5.9-A illustre l'utilisation du niveau supérieur du système de cache par le rendu volumique préintégré. Le rendu volumique de faibles valeurs de l'attribut de semblance met ici en évidence la présence de failles. De même, l'image de la Figure 3.16 du Chapitre 3 représentant le résultat du rendu volumique préintégré avec illumination locale sur les données sismiques est extraite du même jeu de données de 5,2 Go. La sonde volumique dans ce cas a une taille de $480\times480\times510$. Dans ces deux scénarios, la sonde visualisée a une taille de l'ordre de 100 Mo qui peuvent donc être intégralement chargés en mémoire graphique. Lors de déplacements de cette dernière par l'utilisateur, le cache est mis à jour automatiquement pour assurer la présence en mémoire graphique des données visualisées.

La Figure 5.9-B concerne le système de visualisation volumique multimodale du Chapitre 4. Les amplitudes sismiques sont plaquées en temps réel sur une surface d'isovaleur non polygonale d'une carte de distance à un ensemble de puits de production. Dans ce cas, l'extension de la sonde est relativement importante par rapport à celle de la Figure 5.9-A, mais l'élimination des espaces vides permet de réduire significativement le nombre de briques effectivement chargées en mémoire graphique et rendues. En effet, la technique des histogrammes binaires (Section 5.4.2) s'étend de manière triviale au cas multimodal.

La Figure 5.9-C illustre un cas concret de couplage des algorithmes de visualisation et d'extraction d'horizons sismiques. Préalablement au calcul d'extraction, une sonde a été positionnée avec rendu volumique préintégré, ce qui a nécessité le chargement de briques dans le niveau supérieur du cache en mémoire graphique sous forme de textures. Ce chargement suppose un passage dans le niveau inférieur. Après les calculs d'extraction qui ont modifié l'ensemble résidant en mémoire centrale, un résidu de ce chargement subsiste sur la sonde volumique.

Pour conclure, la Figure 5.9-D illustre le modèle structural construit à partir du jeu de données décrit au début de cette section. Les horizons ont été extraits par l'algorithme semi-automatique évoqué à la Section 4.3.1. Les briques utilisées ont une taille homogène dans l'espace de 64^3, ce qui assure une fréquence de défauts de cache optimale à en juger par les courbes de la Figure 5.6. En pratique, les temps d'extraction sont de l'ordre de 1 minute pour chaque horizon. Ainsi, l'extraction des sept horizons visibles sur le modèle nécessite de l'ordre de 10 minutes, ce qui reste très raisonnable sur un volume de plusieurs gigaoctets de données considérant d'autre part que, sans le système de cache, tous les accès aux données se font sur disque et que 10 minutes est l'ordre de temps nécessaire pour extraire chaque horizon individuellement. Un certain nombre de post-traitements sont cependant nécessaires mais n'impliquent pas l'utilisation de notre système, de même que l'extraction des failles qui repose sur des outils spécialisés du logiciel Gocad indépendants de notre travail.

5.6 Bilan et perspectives

En résumé, le système de cache que nous avons présenté au cours de ce chapitre peut paraître relativement limité du point de vue de la visualisation seule. En effet, les techniques de rendu volumique depuis l'espace de compression [Schneider J. et Westermann R., 2003] par exemple semblent beaucoup plus évoluées et prometteuses pour la visualisation de grande quantités de données. Cependant, leur caractère extrêmement spécialisé qui fait leur force constitue également une faiblesse pour des cas d'applications industrielles plus larges qui font intervenir un cahier des charges beaucoup plus discriminant. Il nous semble que la force de notre système est au contraire son caractère générique qui assure son fonctionnement dans des scénarios complexes d'utilisation faisant intervenir toutes sortes de situations de visualisation et de calculs avec des accès indifféremment en lecture et en écriture aux données.

Nous sommes cependant conscients des limitations qu'impose la contrainte de couplage visualisation/calcul dans la mise au point d'un système qui puisse exprimer pleinement son potentiel sur les cas de pure visualisation. Au cours de nos diverses expériences d'interprétation, il est apparu que le présent système de cache reste parfaitement viable pour les calculs même sur de très grandes quantités de données de l'ordre de plusieurs dizaines de gigaoctets, l'extraction d'horizon par exemple tirant pleinement parti du chargement en mémoire centrale de petites briques dans la zone d'extraction. En revanche, dès lors qu'il s'agit de visualiser les données, la taille du domaine d'intérêt que l'utilisateur va chercher à déplacer dans sa sonde peut rapidement devenir prohibitive. En effet, celle-ci se situe aisément aux alentours du gigaoctet de données volumiques que le système tel qu'il a été présenté ne permet pas de gérer sur une station d'interprétation standard.

Après avoir introduit les notions essentielles de gestion de la mémoire sur un système de cache relativement simple pour station d'interprétation, nous allons maintenant pousser ces concepts pour mettre en œuvre un système de cache spécialisé beaucoup plus complexe dont l'objectif est d'assurer la visualisation et le déplacement en temps réel de sondes volumiques de grande taille (1-2 Go) dans des volumes de données de très grande taille (100-200 Go). Ce problème particulièrement exigeant ne constitue pas le quotidien de l'interprétation sismique, aussi nous considérons que l'utilisateur est prêt à recourir ponctuellement à un cluster graphique. C'est ainsi que nous quittons le contexte de la station d'interprétation sismique dans lequel nous nous étions placés en introduction de la Partie II.

Troisième partie

Le cluster graphique

Introduction au cluster graphique

Après avoir introduit diverses techniques de visualisation de données volumiques dans le contexte de l'interprétation sismique sur une station de travail standard, la Partie II a été l'occasion de détailler la mise en œuvre d'un mécanisme de gestion de la mémoire par un système de cache capable de supporter simultanément des situations de visualisation et de calculs. Il est cependant apparu que ce système de cache, par manque de spécialisation, présente un certain nombre de limites dès lors qu'il s'agit de visualiser des volumes de plusieurs dizaines de gigaoctets et au-delà. Or, ce type de situation intervient fréquemment dans les premières étapes de l'interprétation sismique. Pour comprendre les aspects fondamentaux du problème ainsi que la démarche que nous avons décidé de suivre pour y apporter des solutions, il est nécessaire de revenir sur la notion d'échelle d'interprétation introduite en Section 1.2.3 lors de la distinction entre macro- et micro-interprétation. Il existe en effet différents niveaux de raisonnement dans la démarche d'interprétation qui interviennent dans des phases distinctes. La phase quantitative de micro-interprétation à échelle réservoir succède généralement à une phase initiale de macro-interprétation à dominante qualitative permettant la reconnaissance du bassin sédimentaire à échelle régionale et ainsi la mise en évidence des zones de réservoirs potentiels. En pratique, ce changement d'échelle se traduit généralement par l'extraction de sous-volumes des données initiales qui sont étudiés par des équipes distinctes [Christie M., 2002, Dopkin D. et James H., 2006], les résultats de ces interprétations indépendantes étant éventuellement corrélés pour revenir au contexte global du bassin en phase terminale de l'interprétation. Concrètement, les grandes campagnes sismiques qui couvraient par le passé plusieurs centaines de kilomètres carrés ont laissé place aujourd'hui à des campagnes régionales qui couvrent plusieurs dizaines de milliers de kilomètres carrés [Brown A.R., 2004, Dopkin D. et James H., 2006]. Ainsi, une campagne de 22500 Km^2 couvre une surface terrestre de 150 Km de côté à une résolution de l'ordre de 30 m. Considérant que l'acquisition en profondeur atteint plusieurs kilomètres avec une précision de l'ordre la dizaine de mètres, les cubes sismiques ainsi générés peuvent avoir une taille de l'ordre de $4000\times5000\times5000$, ce qui représente 100 Go pour des données codées sur 8 bits. À titre de comparaison, le volume utilisé dans le cas d'étude du Chapitre 5 avait une taille de $2608\times661\times811$, ce qui, pour des données 8 bits, représente environ 1 Go, soit une taille cent fois inférieure. Il s'agit en fait d'une échelle intermédiaire entre l'échelle régionale et l'échelle réservoir où les cubes ont une taille de l'ordre de $300\times400\times400$, soit environ 50 Mo. Ces ordres de grandeur sont résumés dans le Tableau 1 qui, outre la taille des volumes impliqués aux différentes échelles, reporte les tailles minimales de sondes qu'il est nécessaire de manipuler pour recueillir une information intéressante. Les 100 Mo de l'échelle intermédiaire du Chapitre 5 représentent déjà 25 fois plus de données qu'une sonde réservoir typique de 4 Mo mais 20 fois moins qu'une sonde régionale de 2 Go. Ainsi, une sonde classique à échelle régionale est 500 fois plus grande qu'une sonde à échelle réservoir, 20 fois plus grande qu'une sonde à échelle intermédiaire et du même ordre de grandeur que le

	Extension réelle (superficie)	Dimensions du volume (taille mémoire)	Dimensions de la sonde (taille mémoire)
Échelle régionale	150 Km × 150 Km (22500 Km^2)	4000×5000×5000 (100 Go)	2000×1000×1000 (2 Go)
Échelle intermédiaire	30 Km × 30 Km (900 Km^2)	2600×600×800 (1 Go)	400×500×500 (100 Mo)
Échelle réservoir	12 Km × 12 Km 144 Km^2	300×400×400 (50 Mo)	100×200×200 (4 Mo)

TAB. 1 – Ordres de grandeur des différentes échelles d'interprétation en sismique. L'échelle intermédiaire correspond au cas d'étude du Chapitre 5. L'extension réelle correspond à l'ordre de grandeur de la surface terrestre couverte par la zone étudiée. Les dimensions des volumes et sondes sont données en nombre de voxels sur chaque axe.

cube sismique intégral à échelle intermédiaire. De manière intuitive, les changements d'échelle correspondant à ces chiffres peuvent être expliqués par la Figure 1. Sur cette figure, la taille des volumes et des sondes est schématisée par l'aire des rectangles correspondant. L'objectif de cette partie est d'apporter une solution pour la visualisation et le déplacement en temps réel d'une sonde dans un volume à échelle régionale. Ceux-ci correspondent respectivement au carré bleu et au rectangle global jaune qui, comparés aux autres rectangles (ceux de l'échelle intermédiaire du Chapitre 5 par exemple), permettent de prendre la mesure de l'ordre de grandeur du problème à résoudre. Pour y parvenir, nous considérons que l'utilisateur dispose d'un cluster graphique, chaque nœud du cluster correspondant à une station équipée avec le même type de matériel graphique que la station d'interprétation de la Partie II.

Contributions

Nos résultats ont fait l'objet de publications dans le domaine de la visualisation scientifique [CASTANIÉ L. *et al.*, 2006]. Cet article décrit plusieurs contributions. Le travail initial consiste à mettre en œuvre un système de visualisation volumique parallèle sur cluster graphique. Pour cela, nous proposons une approche par décomposition d'objet en utilisant la librairie DViz de composition d'image développée à l'INRIA Lorraine [CAVIN X. *et al.*, 2005] dont nous avons exposé les grandes lignes dans la dernière section du Chapitre 2. Celle-ci est associée au système de cache du Chapitre 5 pour aboutir à un système de visualisation de données volumiques massives par déplacement de sonde avec gestion de buffers en mémoire centrale et en mémoire graphique. Ce système repose sur un parallélisme de contrôle et des accès asynchrones aux données qui permettent de superposer l'exécution des tâches les plus coûteuses.

L'abstraction du mécanisme d'accès aux données nous permet ensuite de remplacer le système de cache du Chapitre 5 par notre propre implantation de mémoire partagée distribuée. Celle-ci repose sur un système de cache hiérarchique distribué baptisé *DHCS* (de l'anglais "Distributed Hierarchical Cache System") [CASTANIÉ L. *et al.*, 2006]. Ce système expose quatre niveaux de chargement des données : mémoire graphique, mémoire centrale locale, mémoire centrale sur les autres nœuds du cluster et disque. Notre implantation a la spécificité d'ajouter deux niveaux de cache, aux deux extrémités de la hiérarchie (mémoire graphique et disque). À notre connaissance, les implantations précédentes de mémoire partagée distribuée pour la visualisation telles que celle proposée dans [DEMARLE D. *et al.*, 2003] ne disposent pas de niveau de cache en mé-

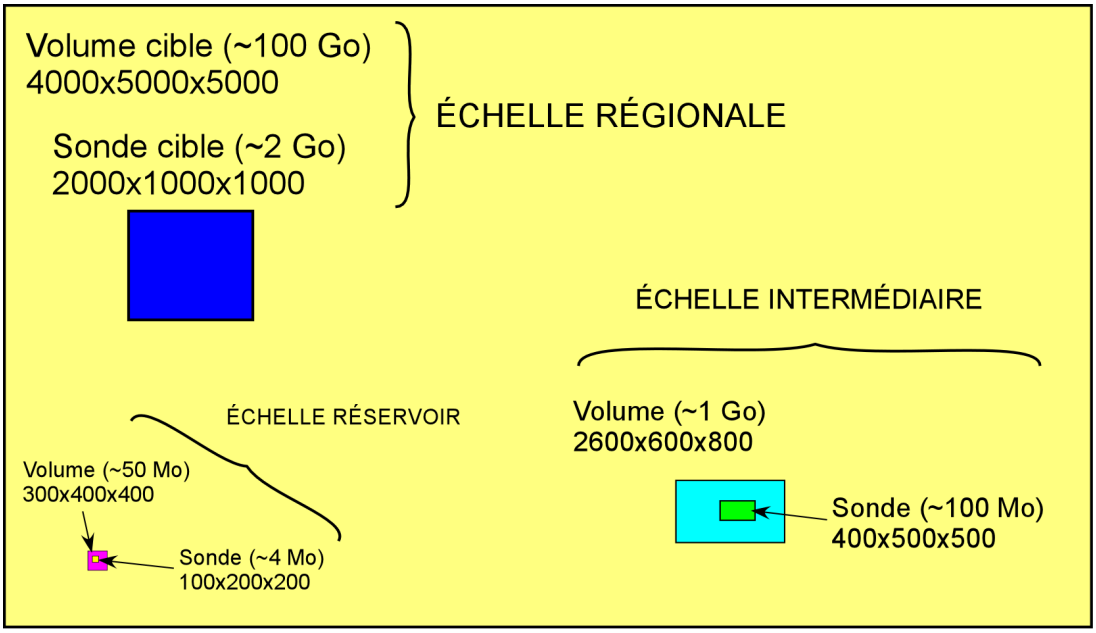

FIG. 1 – Représentation schématique intuitive des différentes échelles d'interprétation des données sismiques. Les tailles relatives des volumes et sondes impliqués sont données par comparaison de l'aire des rectangles correspondant. Ainsi, le carré bleu correspondant à l'objectif à visualiser de sonde à échelle régionale a une aire 20 fois plus grande que le rectangle vert et 500 plus grande que le petit carré jaune, respectivement les sondes à échelle intermédiaire et réservoir. Le rectangle global correspond au volume cible total à échelle régionale.

moire graphique. D'autre part, une différence plus importante encore est que ces implantations n'accèdent pas au disque. Elles disposent de buffers statiques en mémoire centrale et ne peuvent pas gérer des volumes de données plus grands que la taille totale de la mémoire du cluster. Nous nous affranchissons de cette limite par l'utilisation de buffers dynamiques en mémoire centrale. Cependant, comme nous le verrons, ceci nécessite l'existence d'un mécanisme efficace de mise à jour de l'état mémoire du cluster sur chacun des nœuds, ce que nous réalisons à l'aide d'un mécanisme "all-to-all" optimisé de communication entre les nœuds du cluster.

Dans le contexte de l'interprétation sismique, le système présenté au Chapitre 6 laisse entrevoir la possibilité d'intégrer un travail d'interprétation approfondi de certaines zones potentiellement intéressantes dès les premières étapes d'interprétation à l'échelle régionale du bassin sédimentaire. En effet, il fournit un outil d'analyse fine des données par visualisation volumique à une échelle bien supérieure à celle du réservoir. Ceci peut par exemple offrir la possibilité, dès les premières étapes, de disqualifier des zones qui apparaissaient potentiellement intéressantes mais ne le sont pas, ou au contraire de renforcer la conviction qu'une zone mérite une étude plus fine à échelle réservoir ou intermédiaire.

Chapitre 6

Mémoire partagée distribuée pour la visualisation de données massives

Ce chapitre aborde la mise en œuvre d'un système de gestion de la mémoire sur cluster de PCs pour la visualisation et le déplacement en temps réel d'une sonde volumique dans un cube sismique à échelle régionale. Comme il a été vu précédemment, les volumes manipulés à cette échelle atteignent des tailles de l'ordre de la centaine de gigaoctets, nécessitant au minimum des sondes qui dépassent le gigaoctet de données. Dans une telle situation, la complexité du problème à résoudre est de deux ordres : d'une part le rendu en temps réel d'une région d'intérêt de plusieurs gigaoctets de données volumiques, d'autre part son déplacement dans un volume de l'ordre de la centaine de gigaoctets. Comme l'a démontré la Section 2.4 du Chapitre 2, l'utilisation de clusters de PCs est une bonne approche pour la résolution de la première de ces deux problématiques, à savoir le rendu de la sonde en temps réel. Une telle architecture multiprocesseur associée à un modèle de parallélisme de données permet en effet de diviser la quantité de travail sur plusieurs unités de traitement. En revanche, considérant que cette sonde fait partie d'un volume de très grande taille, son déplacement est beaucoup plus problématique. En effet, nous démontrerons dans la section introductive de ce chapitre que, dans ce contexte, l'utilisation d'un système de gestion de la mémoire simple tel que celui introduit au Chapitre 5 ne permet pas d'assurer la réponse instantanée du système lorsque l'utilisateur déplace la sonde. Nous présenterons donc un certain nombre de techniques de gestion de la mémoire plus évoluées dites de mémoire partagée distribuée et mises en œuvre sur des clusters de PCs. Ces techniques destinées à la résolution de problèmes précis présentent cependant un certain nombre de limitations dans le contexte qui est le nôtre. Le cœur de ce chapitre concernera donc les détails d'implantation d'une mémoire partagée distribuée plus évoluée et mieux adaptée au problème exposé plus haut, reposant sur un système de cache hiérarchique distribué baptisé DHCS [Castanié L. *et al.*, 2006].

6.1 Visualisation volumique parallèle et système de cache local

Cette section se concentre sur la conception globale d'un système de visualisation volumique parallèle sur cluster de PCs. Elle fait appel aux notions introduites en Section 2.4 sur l'utilisation du parallélisme en rendu volumique. L'objectif est de mettre en évidence les limitations du système de cache du Chapitre 5 dans ce contexte.

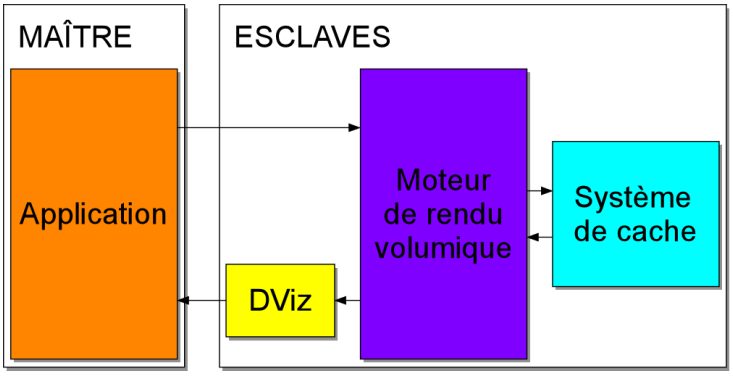

FIG. 6.1 – Principaux composants du système de rendu volumique parallèle sur cluster de PCs et leur interaction.

6.1.1 Visualisation volumique parallèle de sondes sur cluster de PCs

Le premier des deux problèmes identifiés en introduction de ce chapitre, à savoir le rendu volumique d'une région d'intérêt de plusieurs gigaoctets de données, ne peut être résolu simplement par l'utilisation d'une station de travail standard telle que celle qui a prévalu tout au long de la Partie II. En effet, ce type de station présente un certain nombre de limitations matérielles, notamment en termes de quantité de mémoire graphique disponible, qui font que la taille limite de données volumiques pouvant être visualisées en temps réel n'est pas infinie. À cette limitation mémoire s'ajoute la capacité de traitement du processeur de fragment de la carte graphique. Malgré son caractère hautement spécialisé, plusieurs "pixel pipelines" opérant en parallèle (Section 2.3.1), son débit atteint rapidement une limite au-delà de laquelle il n'assure plus les 10-15 fps[19] nécessaires à une manipulation fluide et confortable du point de vue. Dans ce cas, nous avons vu que le recours à une architecture multiprocesseur est une solution naturelle qui permet de réduire la taille du problème d'un facteur d'autant plus grand que le nombre de processeurs impliqués est élevé.

Le système de visualisation volumique parallèle sur architecture multiprocesseur mis en place dans le cadre de ce mémoire repose sur un partitionnement d'objet pour mémoire distribuée de type cluster de PCs. Les implications d'une telle approche ont été exposées en Section 2.4.3. Dans le contexte que nous avons décrit plus haut, une telle approche suppose une décomposition de la sonde volumique en sous-parties dessinées en parallèle sur les nœuds du cluster à la même résolution que l'image finale (Figure 2.16). Ceci suppose l'utilisation d'un algorithme efficace de composition des différentes images obtenues sur chacun des nœuds. Nous utilisons à cette fin la librairie DViz développée à l'INRIA Lorraine [CAVIN X. *et al.*, 2005] qui, comme nous l'avons vu, implante une version extrêmement performante du "binary swap" [MA K.L. *et al.*, 1994]. Les divers composants du système et leur interaction sont illustrés sur la Figure 6.1. Un programme parallèle est décomposé en un ensemble de *processus* constituant les tâches unitaires de traitement. Parmi ces processus, l'un est dit le processus *maître* dont le statut particulier lui incombe la responsabilité de coordonner l'ensemble des autres processus dits *esclaves*. Concrètement, dans le cas d'un cluster de PCs, le processus maître peut être exécuté sur un nœud du cluster ou sur une machine distante reliée au cluster. Les esclaves sont pour leur part exécutés sur les nœuds. Comme le montre la Figure 6.1, l'application, c'est-à-dire le logiciel d'interprétation/visualisation (VolumeExplorer par exemple), correspond au processus maître. Les esclaves quant à eux sont

[19] de l'anglais "frames per second" (images par seconde)

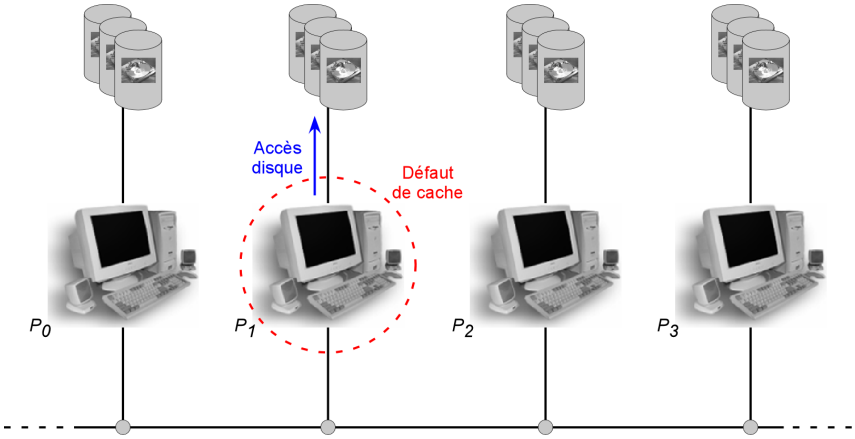

FIG. 6.2 – Principe de fonctionnement d'un système de cache classique local sur station de travail dans le contexte d'un programme parallèle exécuté sur un cluster de PCs. Un défaut de cache en mémoire centrale sur le nœud P_1 (cercle en pointillé rouge) est résolu par un accès disque direct indépendamment de l'état mémoire du cluster (flèche bleue). La représentation sérialisée des liens entre les nœuds est purement conceptuelle et ne traduit en aucun cas le schéma physique d'interconnexion qui existe.

chargés du rendu en parallèle des données et reposent essentiellement sur trois éléments : le moteur de rendu volumique, le système de cache et le compositeur DViz. En pratique, l'utilisateur interagit exclusivement avec l'application sur le processus maître. Cette interaction se traduit par exemple par un changement de point de vue ou un déplacement de la sonde dans le volume. À chaque modification, l'application émet la nouvelle position de la caméra virtuelle (observateur en informatique graphique) ainsi qu'une description des sous-parties de la sonde vers chaque moteur de rendu sur les processus esclaves. Ainsi, chacun dispose des éléments nécessaires au calcul de l'image de sa sous-partie et transmet l'information à son système de cache responsable du chargement des données en mémoire graphique. Le moteur de rendu à base de textures 3D décrit au cours de la Partie II peut alors s'exécuter. Les images des sous-parties produites en parallèle par les esclaves sur les nœuds du cluster sont ensuite composées par le "binary swap" implanté dans DViz pour fournir l'image finale au maître qui la dessine sur l'écran par lequel interagit l'utilisateur. Le présent chapitre porte sur l'implantation d'un système de cache répondant au cahier des charges défini en introduction.

6.1.2 Limitations d'un système de cache local

Tant que l'utilisateur se contente de modifier le point de vue, le système parallèle qui vient d'être décrit suffit à assurer un rendu en temps réel, dans la limite des capacités de la carte graphique présente sur chaque nœud (quantité de mémoire disponible et puissance de traitement du processeur de fragment). Cependant, dès lors que la sonde est déplacée vers une zone encore inexplorée du volume, le système de cache entre en jeu. L'utilisation du système décrit au Chapitre 5, bien qu'envisageable, conduit en pratique à des performances désastreuses. En effet, la Figure 6.2 illustre le comportement du niveau inférieur du cache en mémoire centrale dans ce contexte de visualisation parallèle sur cluster. Un défaut de cache en mémoire graphique dans le niveau supérieur du cache est propagé au niveau inférieur qui, s'il n'est pas capable de le résoudre, accède directement au disque dont la faible bande passante (de l'ordre de 50 MB/s, Chapitre 5) ralentit le système. Lors de déplacements rapides de la sonde par l'utilisateur, ce scénario devient récurrent et le temps de production d'une image est dominé par le temps d'accès

aux données (Section 6.5.2). Comme nous l'avons vu au Chapitre 5, une solution pour réduire le temps de chargement est le recours à des techniques de décimation, lesquelles sont mal supportées par les données sismiques. Il est donc nécessaire de réduire le temps de chargement tout en assurant la présence en mémoire des données à résolution initiale. Comme le montre la Figure 6.2, chaque nœud du cluster exécute en parallèle son propre moteur de rendu volumique sur une sous-partie de la sonde. Lors d'un défaut de cache en mémoire centrale sur un nœud, un autre nœud peut donc potentiellement disposer de la donnée nécessaire. Il est par conséquent possible d'exploiter la bande passante du réseau pour y accéder plus rapidement, celle-ci étant quatre à cinq fois plus grande que celle du disque (de l'ordre de 220 MB/s, Section 6.4.2). C'est ce que se proposent de faire les techniques de mémoire partagée distribuée dont nous allons donner un aperçu à la section suivante.

6.2 Mémoire partagée distribuée pour la visualisation : état de l'art

Dans le cadre de la visualisation, l'objectif des systèmes de mémoire partagée distribuée est d'assurer un partage rapide des données entre les nœuds de rendu à travers le réseau d'interconnexion d'un cluster de PCs. Ceux-ci s'inspire de systèmes généralistes introduits au milieu des années 1980 [Li K., 1986] pour étendre le concept d'espace de mémoire partagé à des architectures multiprocesseurs à mémoire distribuée telles que les clusters de PCs (Section 2.4.2). Un des grands problèmes de ces systèmes de mémoire partagée distribuée généraliste est le maintien de la cohérence des caches lors d'accès en écriture à la mémoire. Sur les SGI Origin introduites en Section 2.4, cette cohérence est assurée par des composants matériels à travers la couche d'interconnexion ccNUMA [Laudon J. et Lenoski D., 1997]. Des recherches intensives ont été menées dans ce sens, conduisant à des implantations logicielles plus ou moins efficaces dont les systèmes Munin [Carter J.B. *et al.*, 1991], Midway [Zekauskas M.J. *et al.*, 1994], Quarks [Carter J.B. *et al.*, 1995] ou encore TreadMarks [Amza C. *et al.*, 1996] sont des exemples. Cependant, dans notre contexte de visualisation, les accès aux données sont en lecture seule, ce qui réduit considérablement la complexité du système en permettant de s'affranchir de ces mécanismes coûteux de maintien de cohérence.

À notre connaissance, la toute première implantation de mémoire partagée distribuée en informatique graphique était destinée à la visualisation parallèle de modèles polygonaux par lancer de rayons avec partitionnement d'image sur cluster de PCs [Green S. et Paddon D., 1989]. Ces travaux précurseurs ont fait l'objet de recherches plus approfondies dans un contexte similaire [Badouel D. *et al.*, 1994] ou dans des systèmes destinés à la visualisation parallèle de données volumiques par lancer de rayons reposant là encore sur du partitionnement d'image [Corrie B. et Mackerras P., 1993, Köse C. et Chalmers A.G., 1997]. Le système le plus récent à notre connaissance a été décliné en deux versions : l'une destinée à la visualisation surfacique [DeMarle D. *et al.*, 2005], l'autre à la visualisation volumique [DeMarle D. *et al.*, 2003]. Dans cette dernière, une décomposition en briques du volume de données introduit la notion de page (Chapitre 5) dans un mécanisme distribué de gestion de la mémoire. La Section 2.4.1 du Chapitre 2 a particulièrement insisté sur l'une des limitations majeures du partitionnement d'image, à savoir le partage de données entre les processeurs qui est peu gênant dans un contexte de mémoire partagée mais problématique dans le cas de mémoires distribuées. L'objectif de ces systèmes est donc double : d'une part il s'agit d'exploiter la cohérence inter-image des tech-

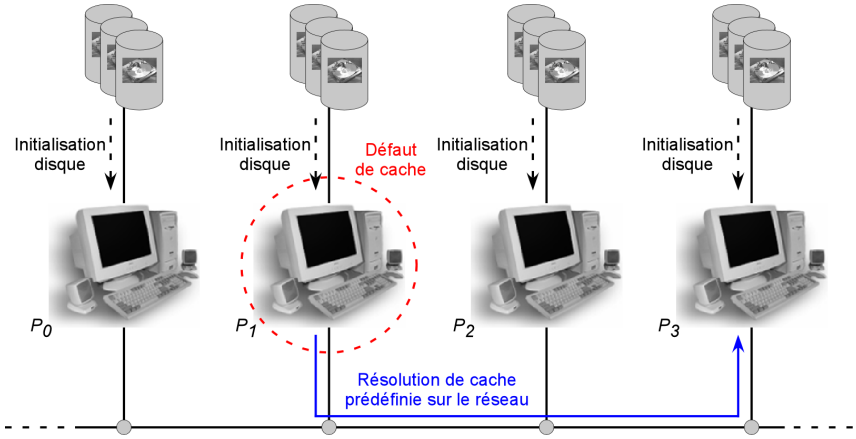

FIG. 6.3 – Principe de fonctionnement d'un système de cache distribué statique sur cluster de PCs. Chaque nœud P_i est initialisé avec un sous-ensemble des données (flèches en pointillés). Un défaut de cache en mémoire centrale sur le nœud P_1 (cercle en pointillé rouge) est résolu de manière prédéfinie par un autre nœud du cluster à travers le réseau d'interconnexion (flèche bleue). La représentation sérialisée des liens entre les nœuds est purement conceptuelle et ne traduit en aucun cas le schéma physique d'interconnexion qui existe.

niques de partitionnement d'image, d'autre part d'assurer un accès rapide aux données à travers le réseau d'interconnexion des nœuds et d'éviter ainsi les accès disques.

Concrètement, l'espace mémoire sur chaque nœud est divisé en deux : un *ensemble résidant* et un *cache*. Au lancement de l'application, les données sont décomposées et les sous-parties sont distribuées statiquement sur les ensembles résidants. La supposition est faite ici que le volume peut être intégralement chargé dans les ensembles résidants des nœuds du cluster. Au moment du rendu, un nœud qui accède une donnée absente de son ensemble résidant et de son cache émet une requête vers le nœud du cluster qui est censé avoir été initialisé avec. Il obtient ainsi la page correspondante qu'il stocke dans son cache pour une utilisation future éventuelle (cohérence inter-image). Ce mécanisme, dont le principe est résumé sur la Figure 6.3, implante en fait un système de cache distribué statique. En effet, l'état des ensembles résidants n'évolue pas au cours du temps. Cette *stratégie de placement directe* [GREEN S. et PADDON D., 1989] simplifie considérablement les requêtes sur le réseau dans la mesure où un défaut de cache sur une page donnée est toujours résolu par le même nœud connu de manière prédéfinie. Cependant, la limitation majeure est que l'ensemble du volume doit tenir dans la mémoire centrale globale du cluster.

Les volumes impliqués dans l'interprétation sismique à échelle régionale dépassent largement la taille globale de la mémoire d'un cluster de PCs standard. Il nous est dès lors nécessaire de proposer un implantation différente de mémoire partagée distribuée. Celle-ci repose en fait sur un système de cache hiérarchique distribué baptisé DHCS dont la conception fera l'objet de la section suivante. Contrairement aux systèmes précédents, l'état de notre DHCS est véritablement dynamique et permet de gérer des volumes dont la taille est uniquement limitée par l'espace disque disponible sur chaque nœud.

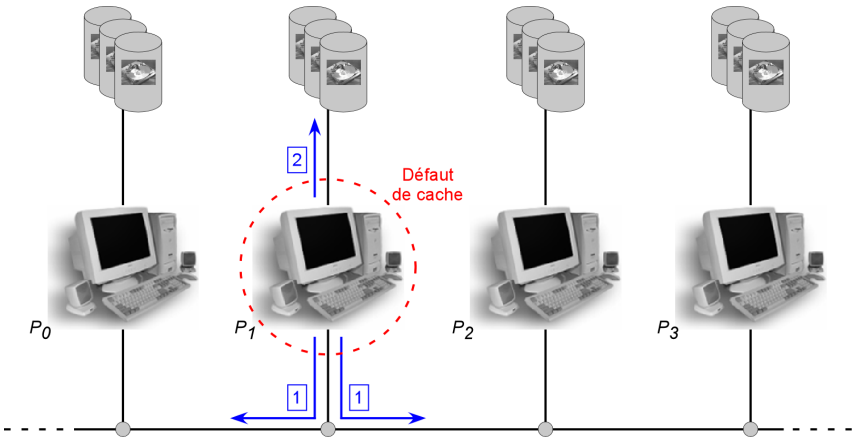

FIG. 6.4 – Principe de fonctionnement d'un système de cache distribué dynamique sur cluster de PCs. Un défaut de cache en mémoire centrale sur le nœud P_1 (cercle en pointillé rouge) est résolu dynamiquement en fonction de l'état mémoire du cluster, en priorité par un autre nœud à travers le réseau d'interconnexion (1), sinon par un accès disque (2). La représentation sérialisée des liens entre les nœuds est purement conceptuelle et ne traduit en aucun cas le schéma physique d'interconnexion qui existe.

6.3 Notre système de cache hiérarchique distribué pour la visualisation

Cette section a pour but de décrire notre système DHCS d'implantation de mémoire partagée distribuée destiné à la visualisation de données sismiques à échelle régionale. Nous en exposerons tout d'abord les grandes lignes à travers les motivations qui ont conduit à sa conception avant de rentrer dans les détails de son fonctionnement.

6.3.1 Motivations

Le système de visualisation volumique parallèle utilisé dans le cadre de ce mémoire repose sur un parallélisme d'objet qui, contrairement au parallélisme d'image, suppose l'absence de partage de données entre processeurs (Section 2.4.1). Cette approche par parallélisme d'objet explique pourquoi le système de cache local décrit au Chapitre 5 suffit à assurer le rendu en temps réel d'une sonde volumique de grande taille (1-2 Go) sur cluster de PCs. En revanche, la Section 6.1.2 a démontré que, dans notre contexte de déplacement de la sonde dans un volume de très grande taille (100-200 Go), la recrudescence des accès disques provoque un effondrement des performances. Or, l'objectif de techniques de mémoire partagée distribuée en visualisation est précisément de s'affranchir des accès disques en les remplaçant par des accès à travers le réseau d'interconnexion des nœuds du cluster. Les précédentes implantations reposent cependant sur un système de cache distribué statique incapable de gérer des volumes de taille supérieure à la quantité de mémoire globale disponible sur le cluster, ce qui les disqualifie dans notre cas de volumes de très grande taille. C'est pour cette raison que nous avons mis au point un véritable système de cache distribué dynamique. Chaque nœud dispose localement d'un cache hiérarchique dont le niveau supérieur est en mémoire graphique. Celui-ci se situe au-dessus d'un niveau complètement dynamique en mémoire centrale. Ainsi, comme le montre la Figure 6.4 pour ce niveau particulier, un défaut de cache provoque une requête sur le réseau dont la résolution par un nœud n'est pas prédéfinie dans la mesure où l'état mémoire du cluster est dynamique.

En cas d'échec de résolution, le chargement de la donnée se fait depuis le disque. De cette manière, les données ne sont pas préchargées statiquement sur les nœuds et la taille du volume global n'est pas limitée par la mémoire du cluster. En revanche, l'état mémoire dynamique de ce dernier implique l'existence d'un mécanisme de mise à jour de répertoires d'état sur chaque nœud permettant l'identification rapide du nœud vers lequel propager un défaut de cache à travers le réseau. Nous reviendrons sur ce point important au cours de la section suivante traitant des détails de conception du système.

6.3.2 Conception du système

Dans le système global de visualisation volumique parallèle sur cluster de PCs dont un aperçu des différents composants apparaît sur la Figure 6.1, le DHCS constitue le système de cache responsable du chargement des données. Comme le montre la Figure 6.6 sur le détail des interactions entre le moteur de rendu et le DHCS sur chaque nœud du cluster, ce dernier comporte deux éléments distincts : une *Hiérarchie de Caches* et un *GPU Loader*. La charte de fonctionnement qui est représentée sur cette figure servira de fil conducteur pour la description du système tout au long de la section. Conceptuellement, il repose sur l'utilisation d'une hiérarchie de buffers, en mémoire centrale et en mémoire graphique, pour exposer un espace virtuel de mémoire graphique au moteur de rendu exécuté sur chaque nœud. Une décomposition du volume en briques de taille unitaire similaire à celle du Chapitre 5 sert de support au mécanisme de pagination à la demande des différents niveaux du cache.

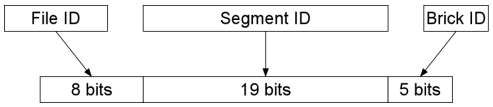

FIG. 6.5 – Principe d'encodage de l'identifiant unique de brique du DHCS dans une clé de 32 bits.

Unité de traitement d'adresses

Notre système DHCS repose sur l'utilisation d'une hiérarchie de buffers : un premier buffer en mémoire graphique sert de cache de la mémoire centrale et un second buffer en mémoire centrale sert de cache d'une série de périphériques (réseau, disque). Les performances des accès à un tel système dépendent de la taille des blocs de données unitaires transférés entre les différents buffers. Comme nous le verrons en Section 6.4, la taille optimale de transfert de données entre disque ou réseau et mémoire centrale est plus grande qu'entre mémoire centrale et mémoire graphique. Pour cette raison, l'élément de taille unitaire manipulé par le système en mémoire centrale est plus grand que celui manipulé en mémoire graphique. La notion de *segment* est introduite en mémoire centrale, un segment correspondant au regroupement de plusieurs briques. Les segments sont ensuite regroupés dans des fichiers stockés sur disque. Ainsi, une brique de données du volume initial est identifiée de manière unique par son fichier d'appartenance, la position de son segment dans le fichier, et finalement sa propre position dans le segment. Comme le montre la Figure 6.5, ces éléments d'identification sont encodés dans une clé de 32 bits. L'identifiant unique qui en résulte sert d'entrée dans des tables de hachage qui stockent l'état de chacun des buffers en mémoire centrale et en mémoire graphique. Ces répertoires d'état constituent de véritables images de l'état des buffers locaux. L'introduction du *File ID* permet la compatibilité avec des systèmes de fichier qui ne supportent pas les fichiers de plus de 4 Go. Dans ce cas, l'identifiant

ainsi défini supporte jusqu'à 256 fichiers, soit 1 To de données. Si au contraire le système de fichiers supporte les fichiers de plus de 4 Go et que, pour des raisons pratiques, il n'est pas possible d'utiliser plus d'un fichier, les 24 bits restants autorisent le stockage de 16 millions de briques, ce qui, pour des briques de 128 Ko, représente 2 To de données. Dans les deux cas de figure, la cible de 100-200 Go de données à échelle régionale reste dans les limites imposées par l'utilisation de cet identifiant.

Le moteur de rendu volumique identifie chaque brique par un triplet d'indices $\{u, v, w\}$ qui décrit sa position dans le volume. Lorsque le DHCS reçoit une requête de la part du moteur de rendu volumique, la première étape consiste à traduire ce triplet en identifiant interne. Sur la Figure 6.6, cette traduction correspond à l'opération *Compute Key* similaire à l'unité de traduction d'adresses d'un système d'exploitation qui traduit les adresses virtuelles en adresses physiques [Silberschatz A. *et al.*, 2005]. À noter cependant que dans le niveau inférieur du cache, en mémoire centrale, la granularité d'une page est plus grande du fait que nous manipulons des segments et non des briques. Dans ce cas, les bits de brique de l'identifiant (*Brick ID*) ne sont pas pris en compte.

Hiérarchie de caches

Le noyau dur du DHCS est constitué par un cache hiérarchique entre mémoire graphique, mémoire centrale, réseau et disque. Dans les systèmes de cache Internet, l'expression hiérarchie de cache [Chankhunthod A. *et al.*, 1996] définit deux types de relations entre les différents caches : une relation verticale *parent-enfant* et une relation horizontale de *frères*. Dans une relation parent-enfant, un défaut de cache dans un enfant est résolu par son parent, plus bas dans la hiérarchie, tandis qu'une relation horizontale de fraternité offre une alternative de résolution par un cache situé au même niveau dans la hiérarchie. Tandis qu'un parent propage un défaut de cache pour le compte de son enfant, la résolution par un frère ne peut être que directe, le défaut de cache ne se propageant pas par ce biais. Les frères sont en fait introduits pour un accès plus rapide par redondance de données sur des caches de même niveau entre lesquels la vitesse de communication est supérieure à celle vers un parent. La résolution horizontale a donc priorité sur la résolution verticale.

Le système de cache introduit au Chapitre 5 repose sur une hiérarchie basique entre mémoire graphique, mémoire centrale et disque. Dans un contexte de visualisation volumique parallèle sur cluster en revanche, chaque nœud du cluster effectue le rendu d'une sous-partie de la sonde volumique et charge ainsi des données dans son espace mémoire local. Ainsi, le déplacement de la sonde peut bénéficier d'un accès plus rapide aux données à travers le réseau depuis les frères situés au même niveau dans la hiérarchie. Le mécanisme de cache est illustré sur la Figure 6.6 à travers les opérations conditionnelles *GPU ?*, *RAM ?* et *LAN ?* qui font respectivement référence à une interrogation des caches en mémoire graphique, en mémoire centrale ou sur le réseau. Il est possible d'identifer une relation parent-enfant entre les caches en mémoire graphique et en mémoire centrale. D'autre part, considérant que chaque nœud exécute son propre DHCS en parallèle, l'opération conditionnelle *LAN ?* fait référence à une requête vers tous les caches en mémoire centrale sur les nœuds du cluster, ce qui illustre une relation de fraternité. DHCS peut donc effectivement être considéré comme un système de cache hiérarchique distribué. Sur chaque nœud, un défaut de cache en mémoire graphique est propagé au cache en mémoire centrale locale. Un défaut de cache en mémoire centrale locale, plutôt que de provoquer un accès disque direct, est propagé aux caches en mémoire centrale sur le cluster (frères). Finalement, un défaut de cache dans tous les frères est résolu par un accès disque.

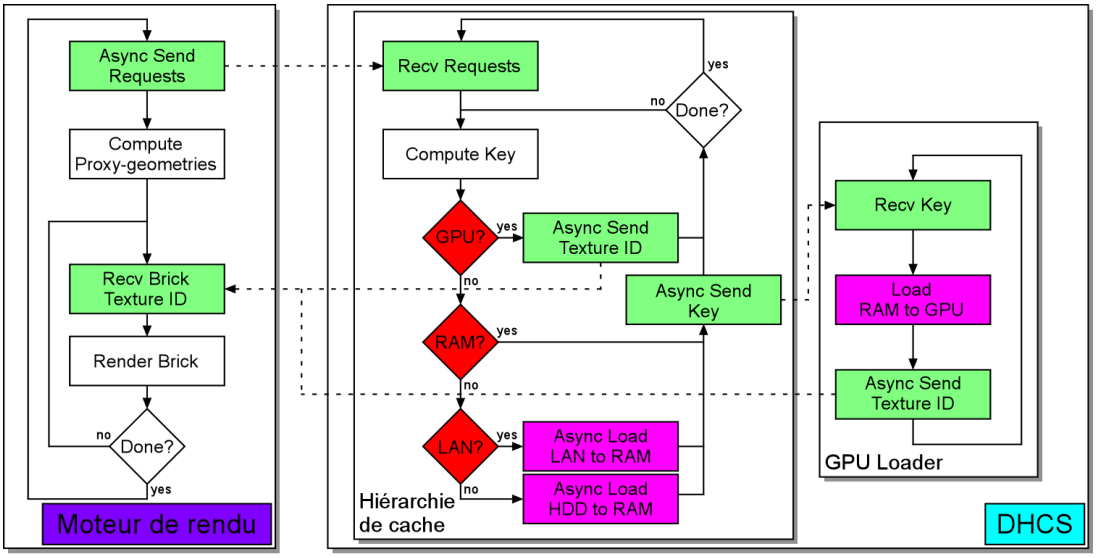

FIG. 6.6 – Principe d'interaction détaillé du moteur de rendu et du système DHCS sur chaque nœud du cluster. Le DHCS se compose de deux éléments distincts : une *Hiérarchie de Caches* et un *GPU Loader*.

Gestion du cache réseau

Dans le contexte de visualisation interactive qui est le nôtre, nous ne pouvons pas nous permettre une propagation vers les frères à travers le réseau comme ceci est fait dans les caches Internet. Au lieu de cela, chaque nœud dispose d'une image de l'état mémoire de l'ensemble du cluster. Dans les implantations précédentes de mémoire partagée distribuée reposant sur un système de cache distribué statique [GREEN S. et PADDON D., 1989, CORRIE B. et MACKERRAS P., 1993, DEMARLE D. *et al.*, 2003], cette image est connue implicitement par les nœuds du fait de la position prédéterminée des pages sur le cluster, ce qui introduit un certain nombre de limitations, notamment l'impossibilité de gérer des volumes plus grands que la quantité totale de mémoire du cluster (Section 6.2). Notre buffer en mémoire centrale est dynamique et nous utilisons un mécanisme de communication "all-to-all" optimisé, dit *"all-to-all" binaire*, qui maintient l'état mémoire du cluster à jour sur chaque nœud. Ce mécanisme de communication s'inspire de l'algorithme du "binary swap" [MA K.L. *et al.*, 1994]. Comme le montre la Figure 6.7 pour le cas de huit nœuds, un tableau contient initialement l'ensemble des clés stockées dans le répertoire d'état local du buffer en mémoire centrale. Puis, chaque paire de nœuds échange un buffer de taille croissante lors d'étapes de communication successives. Au terme de ces échanges, chaque nœud dispose d'un tableau qui contient l'ensemble des clés stockées dans la mémoire du cluster, lui permettant de constituer un répertoire d'état global, véritable image de la mémoire du cluster. Ce mécanisme de communication par échanges binaires successifs est similaire aux opérations "allgather" par *"recursive doubling"* utilisées dans de nombreuses implantations de MPI [THAKUR R. et GROPP W., 2003]. La seule différence significative de notre approche est l'absence de réordonnancement des buffers sur chaque nœud, ce qui nous affranchit d'un certain nombre d'opérations de copie mémoire à chaque étape de communication.

La borne inférieure théorique pour le temps d'exécution d'une communication "all-to-all" est :

$$T_{all-to-all} = \frac{(P-1)\ S}{B} \tag{6.1}$$

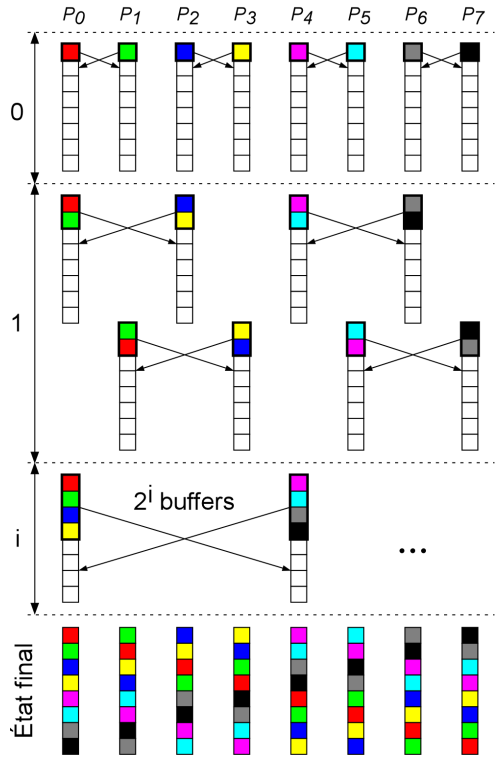

FIG. 6.7 – Principe de fonctionnement du mécanisme de communication *"all-to-all" binaire* dans le cas de huit nœuds P_i.

où P est le nombre de nœuds, S la taille du buffer à échanger et B la bande passante de communication entre les nœuds. Dans notre cas, nous transférons le répertoire d'état de la mémoire centrale locale sous forme d'un tableau de clés. Le temps ainsi calculé est un temps théorique sans latence de communication. Sur un réseau, la latence est définie comme le délai minimum nécessaire à un paquet pour transiter de la source vers la destination indépendamment de sa taille. En pratique, plus le nombre d'étapes de communication est grand, plus l'algorithme est sensible à la latence. L'approche classique consiste à mettre en œuvre une *boucle logique* sur laquelle les nœuds s'échangent leur buffer sur une boucle de communication. Une telle stratégie a une complexité $P - 1$ en termes de nombre d'étapes de communication, ce qui conduit au temps d'exécution théorique suivant :

$$T_{boucle\_logique} = \frac{(P-1)\ S}{B} + (P-1)\ L \tag{6.2}$$

où L est la latence du réseau d'interconnexion des nœuds. La taille des buffers transférés à chaque étape de notre mécanisme de communication "all-to-all" binaire croît, ce qui réduit la complexité à $\log(P)$ et conduit au temps d'exécution théorique suivant :

$$T_{binaire} = \frac{(P-1)\ S}{B} + \log(P)\ L \tag{6.3}$$

où log est le logarithme en base 10. Le principal avantage de ce mécanisme binaire par rapport à une boucle logique classique est son passage à l'échelle du fait de sa complexité en $\log(P)$.

D'autre part, comme nous allons le voir dans la section suivante, ces étapes de communication sont exécutées en parallèle des calculs de rendu sur le processeur graphique, ce qui permet d'en réduire d'autant plus l'impact réel sur les performances du système.

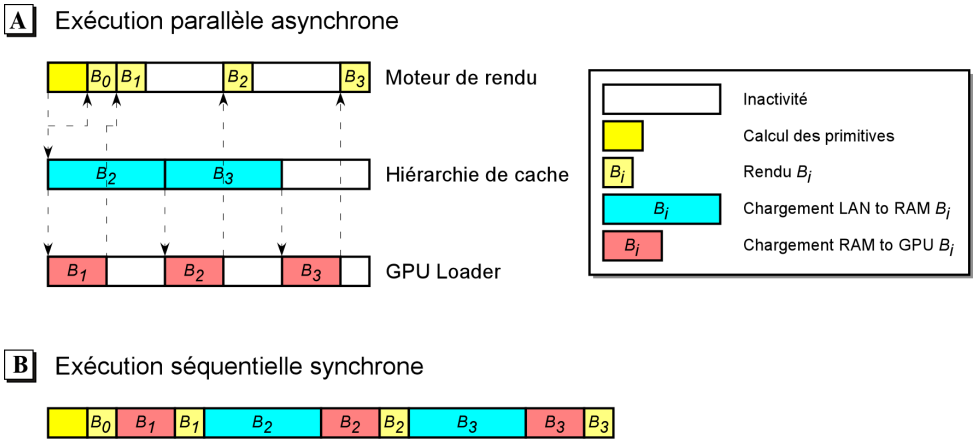

FIG. 6.8 – Comparaison schématique des temps de rendu de quatre briques B_i sur un nœud du cluster dans un modèle d'exécution parallèle avec chargements asynchrones (A) et dans un modèle d'exécution séquentielle avec chargements synchrones (B). Initialement, la brique B_0 est supposée résidante en mémoire graphique, la brique B_1 en mémoire centrale et les briques B_2 et B_3 en mémoire centrale sur d'autres nœuds du cluster.

6.3.3 Implantation d'un parallélisme de contrôle local

Dans cette section, nous nous concentrons sur les composants du système global de la Figure 6.1 impliqués directement dans le rendu local sur chaque nœud : le moteur de rendu et le système de cache DHCS, dont les détails de fonctionnement sont décrits à la Figure 6.6. Sur cette dernière apparaît d'autre part la superposition dans le temps des étapes de transfert de données et du rendu à proprement parler des briques, ce qui permet de masquer les temps de transfert entre les différents buffers. Nous adoptons deux approches complémentaires pour la superposition d'opérations : le parallélisme de contrôle et les chargements asynchrones.

Chaque composant du système, le moteur de rendu et le DHCS, est implanté dans un processus séparé. Lorsque le moteur de rendu émet une requête vers le DHCS pour le chargement d'une série de briques en mémoire graphique, tous les niveaux du cache sont parcourus et les chargements nécessaires sont initiés de manière asynchrone. *Async Load HDD to RAM* fait référence au chargement depuis le disque vers la mémoire centrale, lequel peut être implanté à l'aide de la librairie *libaio* sur Linux qui permet les entrées-sorties asynchrones sur fichiers. *Async Load LAN to RAM* fait référence pour sa part au chargement à travers le réseau qui est implanté dans un processus séparé utilisant un protocole client-serveur avec une couche minimaliste au-dessus du protocole TCP. Finalement, le chargement des textures en mémoire graphique est pris en charge dans un processus supplémentaire qui exécute le composant *GPU Loader*. Nous utilisons la librairie de programmation graphique OpenGL qui ne permet pas à un processus de charger des textures directement dans le contexte graphique associé à un autre processus. Ainsi, le processus qui exécute le composant *GPU Loader* ne peut pas charger des textures directement dans le contexte graphique principal associé au processus qui exécute le moteur de rendu. Nous contournons ce problème par l'utilisation, dans le processus de chargement, d'un contexte graphique indépendant mais créé avec l'option de partage des ressources activée, laquelle autorise le chargement et l'utilisation de ressources (textures, ...) par deux processus différents.

Tandis que le DHCS parcourt la hiérarchie de caches et lance les chargements de données asynchrones, le moteur de rendu calcule les primitives géométriques nécessaires à l'échantillonnage des briques de textures 3D. Dès qu'une brique est chargée dans le niveau supérieur du cache

en mémoire graphique, elle est dessinée en prenant garde de respecter l'ordre de leur composition de l'arrière vers l'avant. Comme le montre schématiquement la Figure 6.8, une implantation séquentielle de ce système avec des requêtes de chargement synchrones, c'est-à-dire bloquantes, allonge significativement le temps nécessaire pour produire une image. Dans ce cas, il s'agit de faire le rendu de quatre briques B_i, B_0 étant résidante en mémoire graphique, B_1 en mémoire centrale, B_2 et B_3 absentes de la hiérarchie de cache locale mais récupérées à travers le réseau. Pour illustrer notre propos, nous imposons arbitrairement que le temps de chargement des briques sur le réseau soit deux fois plus grand que vers la mémoire graphique, qui soit par ailleurs égal au temps de rendu. Dans un modèle d'exécution séquentielle synchrone, les chargements de données à chaque niveau du cache, le calcul des polygones d'échantillonnage et le rendu à proprement parler sont exécutés en série, ce qui, dans cet exemple, double le temps nécessaire pour produire l'image.

D'autre part, dès lors que le DHCS a initié tous les chargements de données asynchrones, l'état local de la mémoire centrale à la fin de l'image courante, à savoir lorsque les chargements seront terminés, est connu. Ceci permet donc de lancer le mécanisme de communication "all-to-all" binaire en parallèle du rendu des briques.

Les détails de fonctionnement du système étant maintenant connus, il est important de préciser que les performances réelles dépendent en grande partie de composants matériels. En effet, un certain nombre de paramètres liés à l'exploitation plus ou moins optimale de ces composants ont une influence significative sur les intervalles de temps passés dans chacune des opérations de la Figure 6.6. Ce sont de ces problèmes de dimensionnement que traite la section suivante.

6.4 Dimensionnement des niveaux de cache

Il existe au sein du système décrit dans la section précédente quelques paramètres sensibles dont l'impact sur les performances globales est significatif. En effet, la taille de page dans chaque niveau du cache est, par exemple, un élément déterminant de la bande passante effectivement exploitée lors d'accès à un périphérique particulier. Il est d'autre part important de préciser que nous disposons ici de toute la latitude nécessaire pour régler ces paramètres en fonction des impératifs de visualisation, latitude qui nous faisait défaut au Chapitre 5 dont le cahier des charges imposait un couplage de la visualisation et des calculs.

6.4.1 Cache en mémoire graphique

Le cache en mémoire graphique est le niveau supérieur de la hiérarchie et, pour cette raison, celui qui supporte le plus grand nombre d'accès aussi bien en écriture qu'en lecture. Il est donc primordial de maîtriser au mieux les paramètres de sa performance. Les accès en écriture lors du chargement des textures en mémoire graphique et les accès en lecture lors du rendu seront successivement abordés.

Chargement des textures

Le chargement des textures concerne les accès en écriture au cache en mémoire graphique et sa performance peut être mesurée par la bande passante effectivement exploitée entre la mémoire

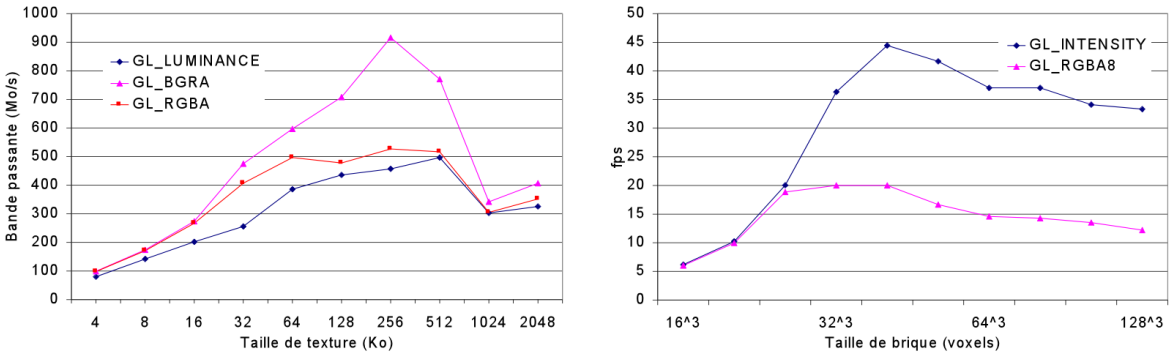

FIG. 6.9 – Performance des accès en lecture/écriture au cache en mémoire graphique (NVIDIA GeForce 6800 Ultra - PCI Express). Gauche : Bande passante effective entre mémoire centrale et mémoire graphique en fonction de la taille de texture pour les formats de texture `GL_LUMINANCE`, `GL_BGRA` et `GL_RGBA`. Le format interne est `GL_INTENSITY` pour `GL_LUMINANCE`, et `GL_RGBA8` pour `GL_BGRA` et `GL_RGBA`. Droite : Taux de rafraîchissement de l'écran en fonction de la taille de brique en voxels pour des formats internes de texture `GL_INTENSITY` et `GL_RGBA8` lors du rendu d'un volume test de 50 Mo.

centrale et la mémoire graphique. Or, celle-ci dépend à la fois du format de texture OpenGL[20] spécifié lors du transfert des données vers la mémoire graphique [NVIDIA CORPORATION, 2005], et de la taille des données transférées. Les courbes de gauche de la Figure 6.9 donnent la bande passante en fonction de la taille et du format de texture lors du chargement de textures 3D. Ces résultat sont intimement liés au matériel graphique utilisé, une carte NVIDIA GeForce 6800 Ultra et un bus PCI Express dans ce cas. La bande passante maximale atteinte est obtenue pour un format de texture `GL_BGRA` et un format interne `GL_RGBA8`. La raison est que, pour les textures 8-bit, les cartes NVIDIA respectent la norme Microsoft GDI de format de pixel qui impose un stockage interne dans un format `BGRA` [NVIDIA CORPORATION, 2005]. Par conséquent, lors du transfert de données qui ne sont pas au format `BGRA`, le pilote de la carte graphique doit intervertir les composants de chaque pixel de la texture, ce qui introduit un surcoût de chargement. Le format interne spécifié lors du chargement `GL_RGBA8` n'a qu'un impact sur le nombre de bits par pixel et non sur l'organisation interne effective des pixels qui respecte toujours la norme GDI. Le second enseignement important des courbes de gauche de la Figure 6.9 est que, pour ce format optimal, la bande passante maximale sur une NVIDIA GeForce 6800 Ultra est obtenue lors du transfert de textures de 256 Ko.

Accès aux textures

Plus que le chargement des textures, la vitesse d'accès aux données une fois que celles-ci sont disponibles en mémoire graphique a un impact significatif sur les performances globales du système. C'est en effet ce second point qui détermine directement le taux de rafraîchissement de l'écran lorsque l'utilisateur se contente de modifier le point de vue sans déplacer la sonde. Or, ceci dépend en grande partie de la taille du cache L1 (cache de texture) sur le processeur graphique, élément sur lequel les fabricants communiquent très peu (Section 5.1.1). Nous étudions l'impact de la taille de brique utilisée sur le taux de rafraîchissement de l'écran pour des formats de texture internes `GL_INTENSITY` et `GL_RGBA8` lors du rendu d'un volume test de 50 Mo. Pour ces tests nous utilisons une post-classification (Section 2.2.2), à savoir que les valeurs scalaires

[20] voir [SEGAL M. et AKELEY K., 2003] pour plus de détails sur les formats de texture OpenGL

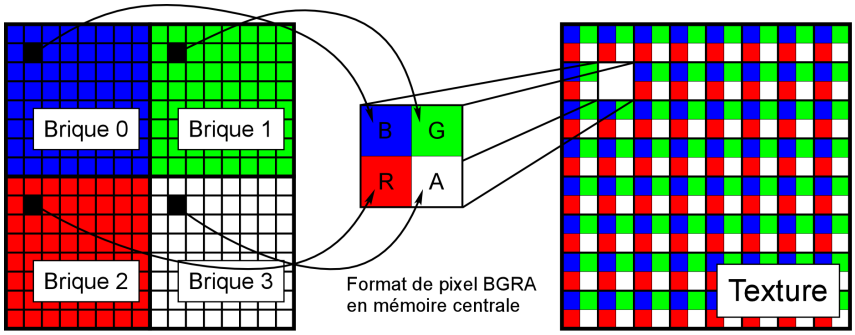

FIG. 6.10 – Principe d'entrelacement des valeurs scalaires de quatre briques dans un seule texture `GL_BGRA` (stockée dans un format interne `GL_RGBA8`).

stockées dans la texture sont converties en émission/absorption par lecture dépendante dans une texture 1D dans un programme de fragment. Dans le cas d'un format interne `GL_RGBA8`, comme le montre la Figure 6.10, nous stockons quatre briques dans une seule texture, chacune d'entre elles dans un canal différent (R, G, B et A). Plutôt que d'introduire des instructions conditionnelles dans un seul programme de fragment, nous utilisons quatre programmes différents pour lire dans l'un des canaux en particulier de telle sorte que le nombre d'instructions soit le même que dans le cas simple d'un format interne `GL_INTENSITY`. Dans le pire des cas, cette stratégie impose un surcoût de permutation de programme de fragment avant le rendu de chaque brique, ce qui d'après nos tests n'a pas d'impact sur les performances. Les résultats apparaissent sur les courbes de droite de la Figure 6.9. Avec des textures `GL_INTENSITY`, le taux de rafraîchissement maximal sur une carte NVIDIA GeForce 6800 Ultra est obtenu pour des briques 64×32×32, ce qui correspond à des textures de 64 Ko. Avec des briques plus petites, le temps de rendu est limité par le processeur central qui calcule les polygones d'échantillonnage, tandis que des briques plus grandes ne tiennent pas dans le cache L1. Dans le cas `GL_RGBA8`, le taux de rafraîchissement maximal, obtenu pour des briques 32×32×32, est deux fois plus lent que pour le cas `GL_INTENSITY`. Le temps de rendu est limité par le processeur central jusqu'à une taille de brique de 32×32×16, laquelle correspond déjà à une taille de texture de 64 Ko du fait que dans ce cas nous stockons quatre briques par texture. Des briques plus grandes produisent en fait des textures plus grandes que le cache L1, ce qui affecte significativement les performances du fait de notre approche par briques entrelacées.

Pour conclure sur cette étude du cache en mémoire graphique, nous utilisons un format `GL_LUMINANCE` et un format interne `GL_INTENSITY` avec une taille de page de 64 Ko (briques de 64×32×32). Le chargement de données en mémoire graphique est plus rapide avec un format `GL_BGRA` et un format interne `GL_RGBA8` (Figure 6.9, courbes de gauche), cependant le format interne `GL_INTENSITY` est deux fois plus rapide lors du rendu (Figure 6.9, courbes de droite). Le gain de performance qu'il implique en termes de vitesse de rendu fait du couple format `GL_LUMINANCE` et format interne `GL_INTENSITY` le meilleur choix, même en considérant les cas d'interaction avec l'utilisateur de type déplacement de la sonde.

6.4.2 Cache en mémoire centrale

La discussion sur le chargement et l'accès aux textures concerne exclusivement le niveau supérieur du DHCS. Nous allons maintenant porter notre attention sur le niveau inférieur en

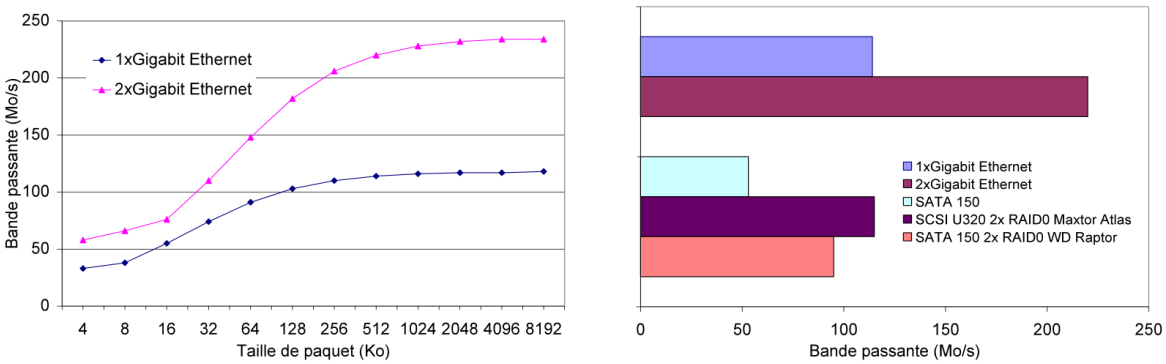

FIG. 6.11 – Comparaison des bandes passantes effectives d'accès à un réseau Gigabit Ethernet et à un disque pour le cache en mémoire centrale. Gauche : Bande passante effective d'interconnexions réseau Gigabit Ethernet en fonction de la taille de paquet pour une et deux interfaces réseau. Droite : Bande passante d'interconnexions réseau Gigabit Ethernet (paquets de 512 Ko) comparée à la bande passante de disques. SATA 150 est notre disque local standard mesuré avec SiSoftware Sandra 2007 [SISOFTWARE SANDRA 2007, –] et les solutions 2x RAID0 sont des configurations RAID0 standards fournies par Sandra à titre de comparaison.

mémoire centrale servant de cache du réseau et des disques. Bien que moins important dans les scénarios de modification du point de vue seul par l'utilisateur où les données sont accédées dans le niveau supérieur en mémoire graphique, ce dernier prend de l'importance lors des déplacements de la sonde, lesquels impliquent en effet des transferts de données à travers l'ensemble de la hiérarchie.

Accès au réseau

Comme le montrent les courbes de gauche de la Figure 6.11, la bande passante réseau effectivement exploitée dépend de la taille des paquets échangés. Plus les paquets sont de petite taille, plus la bande passante est sensible à la latence des communications. Si nous utilisions la même taille de page en mémoire centrale qu'en mémoire graphique (64 Ko), les performances du transfert de données à travers le réseau s'en trouveraient significativement affectées, dans la mesure où nous transférerions des paquets non optimaux de 64 Ko. Pour cette raison, nous avons introduit la notion de segment en mémoire centrale (Section 6.3.2), un segment correspondant au regroupement de plusieurs briques et constituant l'élément de taille unitaire manipulé par le niveau inférieur du cache. Pour un comportement optimum du DHCS, nous utilisons des segments de 512 Ko (2×2×2 briques), ce qui conduit à une bande passante effective de 220 Mo/s dans le cas de deux interfaces réseau Gigagit Ethernet. L'utilisation de segments plus grands augmenterait inutilement le volume des communications au vu du gain de bande passante qu'il offrirait.

Accès au disque

Nous comparons ici la bande passante du réseau pour des pages en mémoire centrale de 512 Ko avec la bande passante de nos disques, des configurations SATA 150 standards. Le résultat apparaît sur le diagramme de droite de la Figure 6.11 qui montre que l'utilisation de deux interfaces Gigabit Ethernet offre une bande passante quatre fois plus rapide que nos disques SATA 150. Il est d'autre part intéressant de relever que la même configuration réseau surpasse

	Gigabit Ethernet	Infiniband 4x	SATA 150
Latence	60 μs	5 μs	10 ms

TAB. 6.1 – Latence d'interconnexions réseau Gigabit Ethernet et Infiniband 4x (d'après [FORCE10 NETWORKS, INC., 2005]) comparées à un disque local standard SATA 150 mesuré avec SiSoftware Sandra 2007 [SISOFTWARE SANDRA 2007, –].

des solutions de disques RAID0 standards. Comme le montrent les courbes de gauche de la Figure 6.11, un autre aspect important lors de transferts de données est la latence. La Table 6.1 rapporte une latence de 10 ms sur nos disques SATA 150 mesurée avec SiSoftware Sandra 2007 [SISOFTWARE SANDRA 2007, –]. Dans le cas de disques, la latence est définie comme étant le temps nécessaire pour positionner la tête de lecture/écriture. Comparée à la latence standard d'interconnexions Gigabit Ethernet de 60 μs [FORCE10 NETWORKS, INC., 2005], il y a un facteur 150x. Ceci est encore plus significatif dans le cas d'interconnexions Infiniband 4x dont la latence est de 5 μs, ce qui introduit un facteur 2000x. Considérant aussi bien la bande passante que la latence, il apparaît clairement que les accès disques ont un impact critique sur les performances de notre système et doivent être évités autant que possible.

Cette section sur l'étude des accès en lecture/écriture aux niveaux du cache en mémoire centrale et en mémoire graphique nous a permis d'identifier les paramètres de la performance de noter système DHCS et d'ajuster leur valeur de manière objective pour un résultat optimal. À partir de ces réglages il va maintenant être possible d'observer le comportement du système de manière globale sur un jeu de données à échelle régionale.

6.5 Expérimentation du système et justification théorique des résultats

Le dimensionnement des paramètres bas-niveau du système permet d'assurer, indépendamment pour chaque composant matériel, des performances optimales. L'objectif de cette section est donc d'observer le comportement global du système dans le temps lorsque tous ces paramètres, ajustés séparément, interagissent les uns avec les autres. Dans un deuxième temps, nous proposerons une justification théorique des résultats observés.

6.5.1 Expérimentation

Nous avons expérimenté notre système sur un cluster de 16 nœuds (bi dual-cœur AMD Opteron 275, 2 Go de RAM, NVIDIA GeForce 6800 Ultra) avec quatre interfaces Gigabit Ethernet par nœud. Comme le montre la Figure 6.12, nous pouvons déplacer une sonde de 1 Go (1024×1024×1024, 8 bits par voxel) dans un volume total de 107 Go (5580×5400×3840, 8 bits par voxel). Pour des raisons de confidentialité, les données sismiques de très grande taille à échelle régionale sont difficiles à obtenir et à montrer publiquement. Pour se placer dans des conditions similaires, nous avons généré le volume de démonstration en assemblant 30 copies du Visible Human [U.S. National Library of Medicine, –], un jeu de données réputé pour la visualisation de données volumiques de grande taille. Le système est capable de maintenir un taux de rafraîchissement de 12 fps en moyenne lors de déplacements dans le cache et avec une distance d'échantillonnage égale à la taille d'un voxel. Nous utilisons, pour cette expérimentation, deux

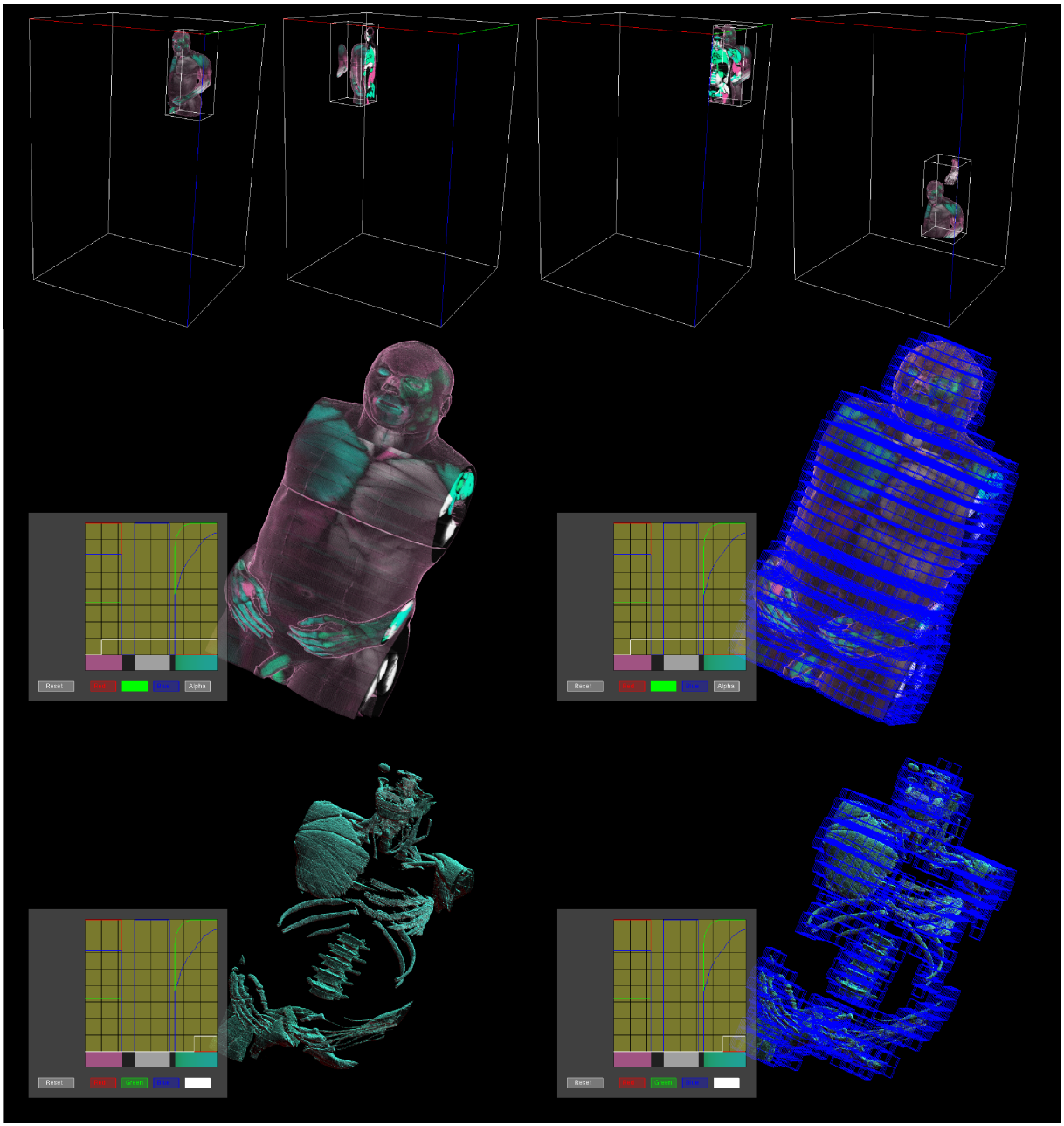

FIG. 6.12 – Visualisation et déplacement d'une sonde de 1 Go (1024×1024×1024, 8 bits par voxel) dans un volume total de 107 Go (30 copies du Visible Human : 5580×5400×3840, 8 bits par voxel) sur un cluster de PCs de 16 nœuds à l'aide du système DHCS. Déplacement en temps réel de la sonde le long des axes principaux du volume à 12 fps en moyenne (haut). Agrandissement sur le rendu volumique préintégré (milieu) avec illumination préintégrée interpolée sur les vertèbres (bas). Les briques effectivement rendues après élimination des espaces vides à l'aide d'opérations bit-à-bit sur des histogrammes binaires sont visibles en bleu (milieu et bas, à droite).

interfaces réseaux pour le compositeur DViz et deux autres pour le système de cache DHCS. D'autre part, la résolution de l'écran est de 1024×768. Enfin, les briques transparentes sont éliminées par l'utilisation d'opérations bit-à-bit sur des histogrammes binaires comme le décrit la Section 5.4.2.

La sonde est déplacée dans chaque direction du volume. Initialement, des manipulations le long des axes bleu et vert permettent de remplir le cache mémoire sur le réseau. Par la suite, lors de déplacements verticaux et en profondeur le long de ces mêmes axes, le taux de rafraîchissement observé est de 12 fps en moyenne. Dans ce cas, la plupart des défauts de cache en mémoire centrale sont résolus depuis un autre nœud du cluster à travers le réseau grâce au DHCS, ce qui évite les accès disques et accélère les temps d'accès aux données d'un facteur 200 en moyenne (section suivante). Cependant, lors de déplacements horizontaux vers la gauche le long de l'axe rouge, l'image se fige régulièrement du fait des accès disques qui interviennent lorsque la frontière du cache réseau est atteinte. Dans ce cas précis, le système bénéficierait grandement de l'utilisation de stratégies de préchargement en fonction du déplacement de la sonde. D'autre part, la décomposition spatiale de la sonde dans notre stratégie de partitionnement d'objet joue un rôle critique sur les performances globales lors des déplacements. En effet, elle influence la taille du front de chargement depuis le disque sur chaque nœud. Nous utilisons une décomposition symétrique qui assure un comportement homogène dans toutes les directions de déplacement.

Finalement, le dernier point important concerne le rendu volumique sur la sonde statique qui dépend uniquement de l'association du compositeur DViz avec notre moteur de rendu volumique. Pour le rendu d'une sonde volumique de 1 Go, nous obtenons un taux de rafraîchissement de 12-13 fps en moyenne qui chute à 8-9 fps lors d'agrandissements sur des points précis de l'image. Il est important de préciser qu'à 13 fps comme à 8 fps nous ne sommes pas limités par le compositeur DViz (dont la limite supérieure à cette résolution est de 45 fps) mais plutôt par la capacité de traitement du processeur de fragment de la carte graphique.

6.5.2 Étude théorique du temps d'accès effectif au DHCS

Partant des observations menées au cours de l'expérimentation précédente, cette étude se propose de fournir une justification théorique des résultats obtenus. L'objectif est d'autre part de mettre en évidence les conditions optimales de comportement du DHCS et de formaliser le gain de performance qui découle d'un tel système de gestion de la mémoire de manière distribuée par rapport à un système local tel que celui du Chapitre 5.

Dans un scénario de visualisation par sonde volumique d'un cube sismique à échelle régionale, le temps d'accès aux données est un élément problématique. En effet, le calcul d'une image pour l'afficher à l'écran se décompose en deux étapes : l'accès aux données, concrètement leur chargement dans les registres du processeur graphique depuis un niveau de sa hiérarchie (Section 5.1.1), et le rendu à proprement parler par le processeur. Ainsi, le temps moyen qui s'écoule entre l'affichage de deux images à l'écran se décompose comme suit :

$$T_{I_{moy}} \;=\; T_{C_{moy}} \;+\; T_{R_{moy}} \tag{6.4}$$

où $T_{C_{moy}}$ et $T_{R_{moy}}$ sont respectivement le temps moyen d'accès aux données à travers le système de cache mis en place et le temps moyen de rendu de l'image.

Indépendamment du système de cache hiérarchique utilisé, local ou distribué, le moteur de rendu volumique implanté dans le cadre de ce mémoire travaille sur des données chargées en mémoire graphique. Dans le cas où celles-ci ne sont pas résidantes en mémoire graphique,

le système de cache est responsable de leur chargement depuis la mémoire centrale. Le *temps d'accès effectif* [Silberschatz A. *et al.*, 2005] à ce cache s'écrit comme suit :

$$T_{C_{moy}} \;=\; T_{GPU} \;+\; m_{GPU}\; T_{m_{GPU}} \tag{6.5}$$

où T_{GPU} est le temps de chargement des données de la mémoire graphique vers les registres du processeur graphique, m_{GPU} la fréquence de défauts de cache en mémoire graphique et $T_{m_{GPU}}$ le temps de résolution d'un défaut de cache en mémoire graphique. Ce dernier dépend d'un second niveau de cache en mémoire centrale, dont le temps d'accès effectif s'écrit de manière similaire :

$$T_{m_{GPU}} \;=\; T_{RAM} \;+\; m_{RAM}\; T_{m_{RAM}} \tag{6.6}$$

où T_{RAM} est le temps de chargement des données de la mémoire centrale vers la mémoire graphique, m_{RAM} la fréquence de défauts de cache en mémoire centrale et $T_{m_{RAM}}$ le temps de résolution d'un défaut de cache en mémoire centrale.

Dans le cas d'un système de cache hiérarchique local comme celui du Chapitre 5, un défaut de cache en mémoire centrale est résolu par un accès disque direct. Le temps de résolution d'un tel défaut de cache est donc identique au temps d'accès disque :

$$T_{m_{RAM}} \;=\; T_{HDD} \tag{6.7}$$

En tenant compte de la bande passante et de la latence des transferts, il est courant de considérer que le temps d'accès à la mémoire centrale est de l'ordre de la centaine de nanosecondes tandis que celui au disque est de l'ordre de la dizaine de millisecondes [Silberschatz A. *et al.*, 2005]. D'autre part, la bande passante entre la mémoire centrale et la mémoire graphique (500 Mo/s, Section 6.4.1) est en pratique dix fois moins grande qu'entre la mémoire centrale et les caches L1/L2 du processeur central (Section 5.1.1), ce qui permet de considérer que le temps d'accès à la mémoire centrale par le processeur graphique est de l'ordre de la microseconde. En considérant finalement que le temps de chargement de la mémoire graphique vers les registres du processeur graphique est négligeable, nous pouvons faire les approximations suivantes :

$$\begin{array}{rcl} T_{GPU} & = & \varepsilon \\ T_{RAM} & = & \tau \\ T_{HDD} & \simeq & 10000\;\tau \end{array} \tag{6.8}$$

où ε est proche de 0 et τ est l'ordre de grandeur du temps de chargement de la mémoire centrale vers la mémoire graphique. Ceci nous conduit à l'expression du temps d'accès effectif à un cache hiérarchique local :

$$\frac{T_{C_{moy}}}{\tau} \;\simeq\; \frac{\varepsilon}{\tau} \;+\; m_{GPU}\;(1 \;+\; 10000\;m_{RAM}) \tag{6.9}$$

Dans l'hypothèse où la sonde peut être intégralement chargée en mémoire graphique, un scénario de visualisation où l'utilisateur se contente de modifier le point de vue n'engendre que des accès au niveau supérieur en mémoire graphique du cache. En d'autres termes, la fréquence de défauts de cache en mémoire graphique est nulle ($m_{GPU} = 0$) et $T_{C_{moy}}$ est négligeable. Dans ce cas, le temps de calcul d'une image est limité par le temps de rendu $T_{R_{moy}}$. Cependant, dès lors que l'utilisateur déplace la sonde, la fréquence de défauts de cache en mémoire graphique tend vers 1. Si le déplacement n'est pas trop grand et que la taille du cache en mémoire centrale est suffisamment grande, la fréquence de défauts de cache dans le niveau inférieur en mémoire centrale reste proche de 0, ce qui conduit à un temps d'accès effectif principalement déterminé par le temps de chargement entre la mémoire centrale et la mémoire graphique :

$$T_{C_{moy}} \;\simeq\; \varepsilon \;+\; \tau \;\simeq\; T_{RAM} \tag{6.10}$$

Ce scénario, viable à petite échelle, devient problématique lorsque la taille des données est très grande et que l'amplitude des déplacements de la sonde dépasse la taille du cache en mémoire centrale. En effet, la fréquence de défauts de cache en mémoire centrale m_{RAM} tend alors vers 1 et le temps d'accès effectif est déterminé par le temps d'accès au disque :

$$T_{C_{moy}} \simeq \varepsilon + 10001\,\tau \simeq T_{HDD} \tag{6.11}$$

Ceci ralentit considérablement l'accès aux données et rend le système de cache du Chapitre 5 impropre à l'interprétation de données sismiques à échelle régionale pour lesquelles ce scénario est courant.

La principale limitation d'un système de cache local vient en fait de la taille du cache en mémoire centrale locale qui s'avère rapidement trop petite pour de très grandes tailles de données. Dans notre cas de visualisation volumique parallèle sur cluster de PCs, l'introduction d'une relation de fraternité (Section 6.3.2) permet de relier l'ensemble des hiérarchies locales à chaque nœud par l'intermédiaire des caches en mémoire centrale. Ceux-ci peuvent en effet, en cas de défaut de cache localement, s'échanger des données à travers le réseau. Cette relation est prioritaire sur l'accès disque et modifie l'expression du temps de résolution d'un défaut de cache en mémoire centrale comme suit :

$$T_{m_{RAM}} = \omega_{LAN}\,T_{LAN} + (1 - \omega_{LAN})\,T_{HDD} \tag{6.12}$$

où ω_{LAN} est le taux de défauts de cache en mémoire centrale résolus à travers le réseau et T_{LAN} le temps de transfert des données entre deux nœuds sur le réseau. L'utilisation de deux interfaces réseau Gigabit Ethernet fournit une bande passante de l'ordre de 220 Mo/s avec une latence de 60 μs (Section 6.4.2). Dans ce cas, nous pouvons faire l'approximation suivante :

$$T_{LAN} \simeq 50\,\tau \tag{6.13}$$

qui va donc nous conduire à l'expression du temps d'accès effectif au DHCS suivante :

$$\frac{T_{C_{moy}}}{\tau} \simeq \frac{\varepsilon}{\tau} + m_{GPU}\,(1 + m_{RAM}\,(10000 - 9950\,\omega_{LAN})) \tag{6.14}$$

Lorsque l'utilisateur sort du cache en mémoire centrale local ($m_{RAM} \rightarrow 1$), le comportement d'un tel système est différent du précédent dans la mesure où le réseau constitue une alternative d'accès aux données qui modifie l'expression du temps d'accès effectif comme suit :

$$T_{C_{moy}} \simeq \varepsilon + \tau\,(10001 - 9950\,\omega_{LAN}) \tag{6.15}$$

Le comportement du DHCS est donc optimal lorsque ω_{LAN} est proche de 1, étant donné que la satisfaction des requêtes n'implique alors plus d'accès au disque :

$$T_{C_{moy}} \simeq \varepsilon + 51\,\tau \simeq T_{LAN} \simeq \frac{T_{HDD}}{200} \tag{6.16}$$

Les approximations théoriques que nous avons faites permettent donc de conclure que, dans ce cas, le temps d'accès aux données est multiplié par un facteur 200. C'est précisément ce scénario qui prédomine lors des déplacements verticaux et en profondeur de la sonde le long des axes vert et bleu dans l'expérimentation précédente. En effet, cette zone-là ayant été précédemment explorée, les nœuds du cluster ont chargé les données correspondantes en mémoire centrale. Par la suite, lors d'un défaut de cache sur un nœud en particulier, il est fort probable qu'un autre

nœud dispose de la page requise, ce qui permet de s'affranchir des accès disques. La quantité totale de mémoire sur le cluster n'autorise cependant pas le chargement de l'ensemble du volume. Lorsque la sonde atteint la frontière du buffer mémoire global du réseau, il devient alors nécessaire d'accéder au disque, ce qui explique, dans l'expérimentation précédente, la chute des performances lors des déplacements le long de l'axe rouge encore inexploré.

En résumé, la principale limitation d'un système de cache hiérarchique local tel que celui du Chapitre 5 est la taille du cache en mémoire centrale, trop petite par rapport à la taille totale du volume de données. Dans un contexte de rendu sur cluster, l'introduction d'une relation de fraternité entre les hiérarchies de cache locales permet d'augmenter virtuellement la taille de ce buffer et donc de réduire le temps d'accès effectif aux données lorsque l'utilisateur déplace la sonde volumique.

6.6 Bilan et perspectives

Le système de cache présenté au Chapitre 5 pour le couplage de la visualisation et des calculs sur grandes quantités de données volumiques en utilisant une station d'interprétation standard a montré des limites pour le passage à une échelle supérieure de visualisation. En effet, dès lors qu'il s'agit de manipuler des cubes sismiques à échelle régionale, de l'ordre de 100-200 Go, le recours à des techniques de visualisation parallèle sur architecture multiprocesseur est apparu indispensable. L'utilisation de la librairie de composition d'image sur cluster de PCs DViz développée à l'INRIA Lorraine [Cavin X. *et al.*, 2005] en association avec notre moteur de rendu volumique nous a permis de mettre en œuvre un système de visualisation volumique parallèle avec partitionnement d'objet sur cluster graphique. Ce système n'a que partiellement résolu les problèmes soulevés par la visualisation de données sismiques à échelle régionale. En effet, bien que parfaitement adapté au problème du rendu en temps réel d'une sonde de l'ordre de 1 Go de données, il s'est avéré incapable d'en assurer le déplacement de manière fluide. La nécessité de mettre en place un mécanisme de partage des données chargées en mémoire centrale entre les nœuds du cluster est apparue comme une nécessité. Nous avons donc proposé une implantation de mémoire partagée distribuée dynamique qui repose sur un système de cache hiérarchique distribué (DHCS). Du point de vue de chaque nœud du cluster qui exécute ce système de cache, il comporte quatre niveaux de chargement des données : mémoire graphique, mémoire centrale locale, mémoire centrale sur les autres nœuds et disque. Une expérimentation sur un volume de très grande taille, suivie d'une étude théorique du temps d'accès effectif au système, a permis de démontrer que l'introduction du lien réseau entre les hiérarchie locales de cache permet d'assurer un déplacement fluide de la sonde volumique dans une zone beaucoup plus étendue que lors de l'utilisation d'un système de cache local.

D'un point de vue technique, un certain nombre d'améliorations de ce système sont envisageables. Par exemple, les transferts de données entre les nœuds sur le réseau occasionnent un volume de communications qui gagneraient dans certains cas à être équilibrées. En effet, notre mécanisme de communication "all-to-all" binaire (Section 6.3.2) permet à chaque nœud du cluster de connaître l'ensemble des données que les autres nœuds sont en mesure de lui fournir. Partant de cette information, nous avons un équilibrage de charge local qui distribue les requêtes équitablement sur le réseau. En revanche, un nœud n'a pas connaissance à priori des requêtes des autres nœuds, condition indispensable pour un équilibrage de charge global. En effet, une telle connaissance lui permettrait de sélectionner la destination de ses propres requêtes en fonction

de celles des autres. En conséquence, un nœud peut potentiellement se retrouver surchargé par les requêtes émanant des autres. Dans le pire des cas où deux nœuds disposent d'une donnée et où tous les autres en ont besoin, le cas de figure où toutes les requêtes sont postées vers un seul des deux nœuds laissant ainsi l'autre au repos est envisageable. Cependant, un tel équilibrage de charge global implique l'introduction d'un échange "all-to-all" supplémentaire avant le lancement des requêtes. Cette communication ne peut être superposée avec le rendu et les chargements comme c'est le cas de la première (Section 6.3.3), ce qui introduit un surcoût non négligeable. Une solution à court terme que nous n'avons cependant pas évaluée serait le recours à des accès disques forcés lorsque le réseau est surchargé, même si un autre nœud pourrait fournir la donnée requise. Un autre point particulièrement important qui mérite d'être abordé est l'implantation d'une stratégie de préchargement des données depuis le disque. En effet, lorsque l'utilisateur déplace la sonde, il est possible d'anticiper le chargement de briques dans le sens du déplacement. Lors des déplacements au-delà de la frontière du cache réseau dans nos expérimentations (Section 6.5.1), les ralentissements liés aux accès disques seraient ainsi atténués.

En conclusion de ce chapitre, il semble important de rappeler le caractère fortement spécialisé du système qui a été mis en place. Son objectif est clairement la visualisation seule des données, condition indispensable pour être capable de supporter les tailles de volumes rencontrées en sismique. Son utilisation est donc destinée, comme nous avons pu la voir, à la première approche qualitative des données à échelle régionale, servant à identifier de potentiels sous-volumes d'intérêt. Ceux-ci sont ensuite interprétés quantitativement par des équipes distinctes après leur extraction à une échelle manipulable sur une station de travail seule. De tels outils de visualisation volumique à très grande échelle permettent cependant d'entamer un travail plus approfondi dans une zone d'intérêt sans pour autant en extraire les données du volume global. Ceci peut par exemple permettre de disqualifier des zones qui apparaissaient potentiellement intéressantes mais ne le sont pas, ou au contraire de renforcer la conviction qu'une zone mérite une étude plus fine à échelle réservoir. Cependant, malgré ces atouts, un tel système ne répond pas à l'ensemble des attentes de géophysiciens/géologues chargés d'interpréter des données sismiques à échelle régionale. En effet, l'utilisation d'outils de visualisation d'ensemble qui donnent une image de la totalité du bassin est indispensable, ce que ne permet pas le déplacement d'une sonde volumique, aussi fluide soit-il. Dans ce cas, l'intérêt se porte sur des outils 2D de visualisation volumique indirecte, comme de simples sections orthogonales aux axes du volume (Section 2.2.1). Bien que très simples à mettre en œuvre sur des volumes de petite taille, elles deviennent extrêmement complexes à déplacer en temps réel sur des cubes sismiques à échelle régionale. Dans ce cas, le recours à des clusters de PCs permet de fournir une solution viable. Des travaux ont été entrepris à l'INRIA Lorraine ayant conduit à un premier démonstrateur capable de déplacer des sections en temps réel dans un volume de 206 Go tout en assurant un taux de rafraîchissement de l'écran de 27 fps pour une résolution d'écran de 1280×1024 avec un cluster de 16 nœuds (bi dual-cœur AMD Opteron 275, 2 Go de RAM, NVIDIA GeForce 6800 Ultra)[21]. La principale limitation est le temps d'accès aux données. Celles-ci sont réparties de manière intelligente sur le cluster et une stratégie de partitionnement d'objet est employée avec la librairie DViz pour composer l'image finale.

[21] http ://www.dviz.fr

Conclusion générale

L'objectif principal du travail présenté dans ce mémoire était de développer des outils de visualisation de données volumiques issues de la sismique réflexion. Dans ce contexte, les principales problématiques concernaient d'une part la mise en place de techniques de visualisation adaptées à la nature des données et à la démarche d'interprétation, d'autre part la prise en compte des limitations matérielles liées à l'environnement de travail en regard de la taille des volumes manipulés. Nous nous sommes efforcés dans la mesure du possible de traiter à la fois les aspects liés au contexte scientifique de la visualisation et ceux liés au contexte d'application, à savoir les géosciences et plus précisément l'interprétation sismique.

Bilan

La première étape de notre travail a consisté à analyser les raisons de la faible popularité des techniques de rendu volumique dans le cadre de l'interprétation sismique, techniques par ailleurs largement adoptées dans d'autres communautés scientifiques, au premier rang desquelles se trouve l'imagerie médicale. Nous avons ainsi mis en évidence un certain nombre de limitations des approches classiques, sources d'incompatibilité avec les caractéristiques propres aux données sismiques et bien connues des spécialistes de la visualisation volumique. Ceci nous a permis de définir les conditions du succès du rendu volumique dans ce contexte, conditions remplies par les approches modernes qui tirent pleinement parti du matériel graphique à disposition. Notre implantation dans le module commercial d'interprétation VolumeExplorer du logiciel Gocad nous a permis de démontrer le potentiel du rendu volumique pour l'analyse de la structure tridimensionnelle des données sismiques.

D'autre part, nous avons également travaillé sur la mise au point d'outils performants capables de combiner en temps réel plusieurs sources d'information. Cet aspect multimodal de l'interprétation a été abordé sous deux angles différents. D'une part, nous nous sommes concentrés sur la combinaison de plusieurs volumes d'information avec l'implantation d'un système générique de visualisation volumique multimodale particulièrement flexible. D'autre part, nous avons considéré le cas des coupes horizon en perspective, ce qui a nécessité le recours à des techniques de visualisation de grandes surfaces combinées à l'utilisation du matériel graphique pour projeter l'information sismique en temps réel sur l'horizon. Dans les deux cas, les outils mis en place permettent de combiner de manière interactive une ou plusieurs sources d'information sismique avec l'information structurale issue de l'interprétation.

Le problème des contraintes matérielles liées à la taille des données, volontairement ignoré dans un premier temps pour aborder l'aspect qualitatif de l'interprétation, a ensuite été abordé en deux temps. Nous avons tout d'abord introduit un certain nombre de concepts de base de gestion de la mémoire à travers l'implantation d'un système de mémoire virtuelle à plusieurs

niveaux capable de gérer des scénarios divers de couplage de la visualisation et des calculs à une échelle intermédiaire d'interprétation entre l'échelle réservoir et l'échelle régionale du bassin sédimentaire. Les notions introduites sur ce système de base pour station de travail seule ont ensuite été étendues à un système distribué de mémoire virtuelle implanté sur un cluster graphique. Celui-ci a permis d'apporter une solution à la visualisation des données sismiques à échelle régionale. Spécialisé pour la visualisation, il permet en effet de déplacer des sondes de grande taille (1-2 Go) dans des volumes de données de très grande taille (100-200 Go).

Perspectives

Dans un avenir proche, il nous semble en premier lieu important d'insister sur la nécessité de proposer des solutions de visualisation intuitives, condition indispensable à leur adoption par le plus grand nombre. Trop souvent, celles-ci restent le produit d'un spécialiste de la visualisation pour un spécialiste de la visualisation. Les difficultés qu'éprouve l'utilisateur à construire des fonctions de transfert adéquates en rendu volumique sont significatives. Il est évident qu'il s'agit d'une des raisons pour lesquelles cette technique est si peu répandue dans la communauté des géophysiciens/géologues. La visualisation de données médicales en est également un bon exemple. En effet, la difficulté de maîtriser la technique y est moins flagrante dans la mesure où les objets sont beaucoup plus familiers et où une fonction de transfert rapidement ébauchée suffit à donner une perception acceptable que le cerveau se charge inconsciemment de compléter. Pourtant, dès lors qu'il s'agit d'observer plus précisément une partie du corps l'utilisateur prend toute la mesure de la difficulté. Il est donc capital de briser le schéma actuel pour que les solutions de visualisation deviennent réellement le produit de spécialistes de la visualisation pour des spécialistes de domaines d'application précis. Ceci suppose une profonde connaissance du domaine d'application que seuls les échanges réguliers et le travail en synergie peuvent apporter.

Les applications que nous avons présentées dans ce mémoire restent localisées à une partie bien précise de la chaîne d'exploration-production détaillée dans la Section 1.2.1. Il serait pourtant intéressant d'étendre l'étude à la visualisation des données volumiques tout au long de cette chaîne. Lors des étapes suivantes, le format de stockage est cependant différent de celui de l'étape de sismique. Il implique en effet des grilles irrégulières, voire de plus en plus souvent des grilles non structurées (Section 2.1). Des travaux ont précédemment abordé la question de la visualisation de ces grilles dans le cadre du projet Gocad [Lévy B. *et al.*, 2001, Caumon G. *et al.*, 2005, Frank T., 2006] et d'autres sont en cours [Buatois L. *et al.*, 2006]. Il serait dans ce contexte intéressant d'introduire un niveau d'abstraction dans le système de cache du Chapitre 6 afin qu'il soit capable de charger dynamiquement des briques de texture indépendamment de la structure de données sous-jacente. Ceci permettrait notamment de tirer directement parti du système générique de visualisation volumique mis en place au Chapitre 4 indépendamment du type de grille. Actuellement, Frank est obligé dans [Frank T., 2006] de recourir à des techniques indirectes de visualisation volumique par extraction de surfaces d'isovaleur pour y plaquer ensuite l'information de couleur issue d'un ou plusieurs volumes supplémentaires.

À l'instar de la partie aval, la partie amont de l'interprétation sismique dans la chaîne d'exploration-production devient également consommatrice de solutions de visualisation. Les performances que permettent d'atteindre les clusters graphiques ouvrent en effet la voie au contrôle qualité en temps réel du résultat des traitements sismiques sur les données fraîchement acquises [Hodgson P. *et al.*, 2005] au point de voir apparaître des systèmes de visualisation sur cluster embarqués directement sur les vaisseaux d'acquisition [Taylor C. et Lima J., 2005]. Le

système DHCS du Chapitre 6, spécialisé pour les tâches de visualisation, s'inscrit parfaitement dans ce contexte où il s'agit d'explorer rapidement de grandes quantités de données volumiques afin d'y observer localement d'éventuelles anomalies. L'une des limitations au déploiement des solutions sur cluster graphique est cependant le coût de développement que cela implique d'un point de vue industriel. En effet, des logiciels comme Gocad représentent un tel volume de code qu'il serait utopique de vouloir porter l'application dans son ensemble sur ce type d'architectures parallèles. La stratégie employée par la société Earth Decision consiste plutôt à développer la notion de *service de visualisation*. Ainsi, une station cliente, indépendante du cluster, exécute l'application métier standard qui se connecte à distance à un cluster fournissant des services de visualisation distribuée avec éventuellement gestion distribuée de la mémoire comme au Chapitre 6. Le démonstrateur de l'INRIA Lorraine pour la visualisation de coupes orthogonales dans des volumes à échelle régionale fait ainsi déjà l'objet d'un prototype de service sur cluster accessible depuis le logiciel Gocad. Il est fort probable qu'à l'avenir cette solution tende à se développer pour assurer l'accessibilité de solutions cluster à une population plus grande d'utilisateurs sans pour autant engendrer des coûts de développement prohibitifs.

Bibliographie

GPGPU. *General-Purpose Computation on Graphics Hardware*. http ://www.gpgpu.org.

Terrain LOD. *Runtime Regular-Grid Algorithms*. http ://www.vterrain.org/LOD/Papers.

Ahrens J. et Painter J.. *Efficient Sort-Last Rendering Using Compression-Based Image Compositing*. In Proceedings of Eurographics Symposium on Parallel Graphics and Visualization, 1998. pp. 145–151.

Amin M., Grama A. et Singh V.. *Fast Volume Rendering Using an Efficient, Scalable Parallel Formulation of the Shear-Warp Algorithm*. In Proceedings of ACM Symposium on Parallel Rendering, 1995. pp. 7–14.

Amza C., Cox A.L., Dwarkadas S. *et al. TreadMarks : Shared Memory Computing on Networks of Workstations*. Computer, 1996, vol. 29, pp. 18–28.

Asirvatham A. et Hoppe H.. *GPU Gems 2 : Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley, 2005. Terrain Rendering Using GPU-Based Geomatry Clipmaps, pp. 27–45.

Aziz K. et Settari A.. *Petroleum Reservoir Simulation*. Applied Science Publishers, 1979. 497 p.

Backus M.M. et Chen R.L.. *Flat Spot Exploration*. Geophysical Prospecting, 1975, vol. 23, pp. 533–577.

Badouel D., Bouatouch K. et Priol T.. *Distributing Data and Control for Ray Tracing in Parallel*. IEEE Computer Graphics and Applications, 1994, vol. 14, n°4, pp. 69–77.

Bahorich M. et Farmer S.. *3D Seismic Discontinuity for Faults and Stratigraphic Features : The Coherence Cube*. The Leading Edge, 1995, vol. 14, n°10, pp. 1053–1058.

Bajaj C.L., Pascucci V. et Schikore D.R.. *Fast Isocontouring for Improved Interactivity*. In Proceedings of IEEE Symposium on Volume Visualization, 1996.

Bakalash R., Kaufman A., Pacheco R. *et al. An Extended Volume Visualization System for Arbitrary Parallel Projection*. In Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware, 1992.

Barnes A.E.. *Shaded Relief Seismic Attribute*. In Society of Exploration Geophysicists (SEG) 72^{th} Annual Meeting, 2002.

Barnes A.E.. *Shaded Relief Seismic Attribute*. Geophysics, 2003, vol. 68, n°4, pp. 1281–1285.

Becker D.J., Sterling T., Savarese D. *et al. BEOWULF : A parallel workstation for scientific computation*. In Proceedings of the International Conference on Parallel Processing, 1995. pp. 11–14.

Behrens U. et Ratering R.. *Adding Shadows to a Texture-Based Volume Renderer*. In Proceedings of IEEE Symposium on Volume Visualization, 1998. pp. 39–46.

Bhaniramka P. et Demange Y.. *OpenGL Volumizer : A Toolkit for High Quality Volume Rendering of Large Data Sets.* In Proceedings of IEEE Symposium on Volume Visualization, 2002. pp. 45–54.

Blinn J.F.. *Models of Light Reflection for Computer Synthesized Pictures.* Computer Graphics, 1977, vol. 11, n°2, pp. 192–198.

Blinn J.F.. *Simulation of Wrinkled Surfaces.* Computer Graphics, 1978, pp. 286–292.

Blinn J.F.. *Compositing : Theory.* IEEE Computer Graphics and Applications, 1994, vol. 14, n°5, pp. 83–87.

Bloomenthal J.. *Polygonization of Implicit Surfaces.* Computer Aided Geometric Design, 1988, vol. 5, n°4, pp. 53–60.

Bolt B.. *Inside the Earth : Evidence from Earthquake.* San Francisco : W.H. Freeman and Co, 1982. 191 p.

Bonham-Carter G.F.. *Geographic Information Systems for Geoscientists : Modelling with GIS.* Pergamon, 1994. 398 p.

Bosquet F. et Dulac J.C.. *Advanced Volume Visualization - New Ways to Explore, Analyze and Interpret Seismic Data.* The Leading Edge, 2000, vol. 19, n°5, pp. 535–537.

Brown A.R.. *Interpretation of Three-Dimensional Seismic Data, Sixth Edition.* Tulsa : AAPG-SEG, 2004. 541 p.

Buatois L., Caumon G. et Lévy B.. *GPU Accelerated Isosurface Extraction on Tetrahedral Grids.* In Proceedings of International Symposium on Visual Computing, 2006.

Cabral B., Cam N. et Foran J.. *Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware.* In Proceedings of IEEE Symposium on Volume Visualization, 1994. pp. 91–98.

Cabral B. et Leedom L.. *Imaging Vector Fields Using Line Integral Convolution.* In Proceedings of ACM SIGGRAPH Conference, 1993. pp. 263–272.

Cai W. et Sakas G.. *Data Intermixing and Multi-Volume Rendering.* In Proceedings of Eurographics/IEEE Symposium on Visualization, 1999. pp. 359–368.

Carter J.B., Bennett J.K. et Zwaenepoel W.. *Implementation and Performance of Munin.* In Proceedings of the 13^{th} ACM Symposium on Operating Systems Principles, 1991. pp. 152–164.

Carter J.B., Khandekar D. et Kamb L.. *Distributed Shared Memory : Where We Are and Where We Should Be Headed.* In Workshop on Hot Topics in Operating Systems, 1995. pp. 119–122.

Castanié L., Bosquet F. et Lévy B.. *Advanced Volume Visualization Techniques for Seismic Interpretation.* In Society of Exploration Geophysicists (SEG) 75^{th} Annual Meeting, 2005.

Castanié L., Bosquet F. et Lévy B.. *Advances in Seismic Interpretation Using New Volume Visualization Techniques.* First Break, 2005, vol. 23, pp. 21–24.

Castanié L., Lévy B. et Bosquet F.. *VolumeExplorer : Roaming Large Volumes to Couple Visualization and Data Processing for Oil and Gas Exploration.* In Proceedings of IEEE Visualization Conference, 2005.

Castanié L., Mion C., Cavin X. *et al. Distributed Shared Memory for Roaming Large Volumes.* IEEE Transactions on Visualization and Computer Graphics (Proceedings of IEEE Visualization 2006), 2006, vol. 12, n°5, pp. 1299–1306.

Caumon G., Lévy B., Castanié L. *et al. Visualization of Grids Conforming to Geological Structures : A Topological Approach.* Computers & Geosciences, 2005, vol. 31, n°6, pp. 671–680.

Cavin X. et Mion C.. *Pipelined Sort-last Rendering : Scalability, Performance and Beyond.* In Proceedings of Eurographics Symposium on Parallel Graphics and Visualization, 2006.

Cavin X., Mion C. et Filbois A.. *COTS Cluster-Based Sort-Last Rendering : Performance Evaluation and Pipelined Implementation.* In Proceedings of IEEE Visualization Conference, 2005.

Chankhunthod A., Danzig P.B., Neerdaels C. *et al. A Hierarchical Internet Object Cache.* In Proceedings of the USENIX Annual Technical Conference, 1996. pp. 153–164.

Chopra S., Zwicker T., Mitchell K. *et al. Why Is It That Volume Interpretation of 3D Seismic Data Is Not Practiced by Seismic Interpreters ?* CSEG Recorder - Expert Answers, 2005, pp. 8–17.

Christie M.. *Thinking Inside the Box.* CSEG Recorder, 2002, pp. 50–57.

Cignoni P., Ganovelli F., Gobbetti E. *et al. BDAM - Batched Dynamic Adaptive Meshes for High Performance Terrain Visualization.* Computer Graphics Forum, 2003, vol. 22, n°3.

Cignoni P., Ganovelli F., Gobbetti E. *et al. Planet-Sized Batched Dynamic Adaptive Meshes (P-BDAM).* In Proceedings of IEEE Visualization Conference, 2003. pp. 147–154.

Cignoni P., Puppo E. et Scopigno R.. *Representation and Visualization of Terrain Surfaces at Variable Resolution.* The Visual Computer, 1997, vol. 13, n°5, pp. 199–217.

Cline D. et Egbert P.K.. *Terrain Decimation Through Quadtree Morphing.* IEEE Transactions on Visualization and Computer Graphics, 2001, vol. 7, n°1, pp. 62–69.

Cohen-Or D. et Levanoni Y.. *Temporal Continuity of Levels of Detail in Delaunay Triangulated Terrain.* In Proceedings of IEEE Visualization Conference, 1996. pp. 37–42.

Corrie B. et Mackerras P.. *Parallel Volume Rendering and Data Coherence.* In Proceedings of ACM Symposium on Parallel Rendering, 1993. pp. 23–26.

Cox M. et Ellsworth D.. *Application-Controlled Demand Paging for Out-of-Core Visualization.* In Proceedings of IEEE Visualization Conference, 1997. pp. 235–244.

Dachille F., Kreeger K., Chen B. *et al. High-Quality Volume Rendering Using Texture Mapping Hardware.* In Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware, 1998. pp. 69–77.

de Boer W.. *Fast Terrain Rendering Using Geometrical MipMapping.* http ://www.flipcode.com/tutorials/geomipmaps.pdf, 2000.

De Floriani L., Magillo P. et Puppo E.. *Building and Traversing a Surface at Variable Resolution.* In Proceedings of IEEE Visualization Conference, 1997. pp. 103–110.

Debrin R., Carpenter L. et Hanrahan P.. *Volume Rendering.* In Proceedings of ACM SIGGRAPH Conference, 1988. pp. 65–74.

DeMarle D., Gribble C., Boulos S. *et al. Memory Sharing for Interactive Ray Tracing on Clusters.* Parallel Computing, 2005, vol. 31, n°2, pp. 221–242.

DeMarle D., Parker S., Hartner M. *et al. Distributed Interactive Ray Tracing for Large Volume Visualization.* In Proceedings of IEEE Symposium on Parallel and Large-Data Visualization and Graphics, 2003. pp. 87–94.

Dopkin D. et James H.. *Trends in Visualization for E&P Operations.* First Break, 2006, vol. 24, pp. 55–59.

Dorn G.A.. *Modern 3D Seismic Interpretation.* The Leading Edge, 1998, vol. 17, n°9, pp. 1262–1270.

Duchaineau M.A., Wolinsky M., Sigeti D.E. *et al. ROAMing Terrain : Real-Time Optimally Adapting Meshes.* In Proceedings of IEEE Visualization Conference, 1997. pp. 81–88.

Durst M.. *Additionnal Reference to Marching Cubes.* In Proceedings of ACM SIGGRAPH Conference, 1988. pp. 72–73.

El-Sana J. et Varshney A.. *Generalized View-Dependent Simplification.* In Proceedings of Eurographics/IEEE Symposium on Visualization, 1999. pp. 83–94.

Elvins T.. *Volume Rendering on a Distributed Memory Parallel Computer.* In Proceedings of IEEE Visualization Conference, 1992. pp. 93–98.

Engel K., Kraus M. et Ertl T.. *High-Quality Pre-Integrated Volume Rendering Using Hardware Accelerated Pixel Shading.* In Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware, 2001. pp. 9–16.

Evans W., Kirkpatrick D. et Townsend G.. *Right Triangulated Irregular Networks.* Algorithmica, 2001, vol. 30, n°2, pp. 264–286.

Falby J.S., Zyda M.J., Pratt D.R. *et al. NPSNET : Hierarchical Data Structures for Real-Time Three-Dimensional Visual Simulation.* Computer & Graphics, 1993, vol. 17, n°1, pp. 65–69.

Ferguson R.L., Economy R., Kelly W.A. *et al. Continuous Terrain Level of Detail for Visual Simulation.* In Proceedings of IMAGE V Conference, 1990. pp. 144–151.

Fernando R. (Ed). *GPU Gems : Programming Techniques, Tips, and Tricks for Real-Time Graphics.* Addison-Wesley, 2004. 765 p.

Fernando R. et Kilgard M.J.. *The Cg Tutorial : The Definitive Guide to Programmable Real-Time Graphics.* Addison-Wesley, 2003. 336 p.

Flinchbaugh F.. *Method and Apparatus for Automatically Producing Representations of Three-Dimensional Horizons from Processed Seismic Data.* American Patent n° 4 633 402 : Texas Instruments Incorporated, 30 December 1986.

Foley J.D., van Dam A., Feiner S.K. *et al. Computer Graphics - Principles and Practice, Second Edition.* Addison-Wesley, 1996. 1173 p.

Force10 Networks, Inc.. *The High Performance Data Center : The Role of Ethernet in Consolidation and Virtualization.* http ://www.force10networks.com/products/pdf/wp_datacenter_convirt.pdf, 2005.

Frank S. et kaufman A.. *Massive Volume Rendering on a Volume Visualization Cluster.* Technical Report, Stony Brook University, 2004.

Frank T.. *Advanced Visualization and Modeling of Tetrahedral Meshes.* Th. Doct.. Institut National Polytechnique de Lorraine, 2006.

Gao J., Huang J., Johnson R. *et al. Distributed Data Management for Large Volume Visualization.* In Proceedings of IEEE Visualization Conference, 2005. pp. 183–189.

Gasparakis C.. *Multi-Resolution Multi-Field Ray Tracing : A Mathematical Overview.* In Proceedings of IEEE Visualization Conference, 1999.

Gerhardt A., Machado M. et Gattass M.. *A Combined Approach to 3D Seismic Data Segmentation and Rendering.* In Brazilian Geophysical Society (SBGf) 6^{th} International Congress, 1999.

Gerhardt A., Machado M., Silva P.M. *et al. Two-Dimensional Opacity Functions for Improved Volume Rendering of Seismic Data.* In Brazilian Geophysical Society (SBGf) 7^{th} International Congress, 2001.

Gerhardt A., Machado M., Silva P.M. *et al. Enhanced Visualization of 3D Seismic Data.* In Society of Exploration Geophysicists (SEG) 72^{nd} Annual Meeting, 2002.

Günther T., Poliwoda C., Reinhard C. *et al. VIRIM : A Massively Parallel Processor for Real-Time Volume Visualization in Medicine.* In Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware, 1994. pp. 103–108.

Goovaerts P.. *Geostatistics for Natural Resources Evaluation.* Oxford University Press, 1997. Applied Geostatistics Series. 483 p.

Gouraud H.. *Continuous Shading of Curved Surfaces.* IEEE Transactions on Computers, 1971, vol. 20, n°6.

Green S. et Paddon D.. *Exploiting Coherence for Multiprocessor Ray Tracing.* IEEE Computer Graphics and Applications, 1989, vol. 9, n°6, pp. 12–26.

Grzeszczuk R., Henn C. et Yagel R.. *Advanced Geometric Techniques for Ray Casting Volumes.* Course, ACM SIGGRAPH, 1998.

Guthe S. et Strasser W.. *Real-Time Decompression and Visualization of Animated Volume Data.* In Proceedings of IEEE Visualization Conference, 2001.

Hadwiger M., Theussl T., Hauser H. *et al. Hardware-Accelerated High-Quality Filtering on PC Hardware.* In Proceedings of Vision, Modeling and Visualization, 2001. pp. 105–112.

Hodgson P., Cunnel C., Krueger A. *et al. Cluster Visualization Rises to the Seismic Processing Challenge.* First Break, 2005, vol. 23, pp. 69–71.

Hoppe H.. *Progressive Meshes.* In Proceedings of ACM SIGGRAPH Conference, 1996. pp. 99–108.

Hoppe H.. *Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering.* In Proceedings of IEEE Visualization Conference, 1998. pp. 35–42.

Houston M.. *Designing Graphics Clusters.* Parallel Rendering Workshop - IEEE Visualization Conference, 2004.

Howard R.E.. *Method for Attribute Tracking in Seismic Data.* American Patent n° 5 056 066 : Landmark Graphics Corporation, 8 October 1991.

Hsu W.. *Segmented Ray-Casting for Data Parallel Volume Rendering.* In Proceedings of ACM Symposium on Parallel Rendering, 1993. pp. 7–14.

Huang J., Shareef N., Crawfis R. *et al. A Parallel Splatting Algorithm with Occlusion Culling.* In Proceedings of Eurographics Symposium on Parallel Graphics and Visualization, 2000.

Humphreys G., Houston M., Ng R. *et al. Chromium : A Stream-Processing Framework for Interactive Rendering on Clusters.* In Proceedings of ACM SIGGRAPH Conference, 2002.

Ikits M., Kniss J., Lefohn A. *et al. GPU Gems : Programming Techniques, Tips, and Tricks for Real-Time Graphics.* Addison-Wesley, 2004. Volume Rendering Techniques, pp. 667–692.

Itoh T. et Koyamada K.. *Automatic Isosurface Propagation Using an Extrema Graph and Sorted Boundary Cell Lists.* IEEE Transactions on Visualization and Computer Graphics, 1995, vol. 1, n°4, pp. 319–327.

Jen D., Parente P., Robbins J. *et al. Image Surfer : A Tool for Visualizing Correlations Between Two Volume Scalar Fields.* In Proceedings of IEEE Visualization Conference, 2004. pp. 529–536.

Johnson C.. *Top Scientific Visualization Research Problems.* IEEE Computer Graphics and Applications, 2004, vol. 24, n°4, pp. 13–17.

Kajiya J.T. et von Herzen B.P.. *Ray Tracing Volume Densities.* In Proceedings of ACM SIGGRAPH Conference, 1984. pp. 165–174.

Kaufman A.. *Voxels as a Computational Representation of Geometry.* In The Computational Representation of Geometry, Course, ACM SIGGRAPH, 1994.

Kaufman A. et Bakalash R.. *Memory and Processing Architecture for 3D Voxel-Based Imagery.* IEEE Computer Graphics and Applications, 1988, vol. 8, n°6, pp. 10–23.

Kaufman A. et Mueller K.. *The Visualization Handbook.* Elsevier, 2005. Overview of Volume Rendering, pp. 127–174.

Keskes N., Zaccagnino P. et Rether D.. *Automatic Extraction of 3D Seismic Horizons.* In Society of Exploration Geophysicists (SEG) 53^{rd} Annual Meeting, 1983.

Kessenich J., Baldwin D. et Rost R.. *The OpenGL Shading Language, Version 1.051.* 3Dlabs, 2003. http ://www.3dlabs.com/support/developer.

Kidd G.D.. *Fundamentals of 3D Seismic Volume Visualization.* The Leading Edge, 1999, pp. 702–710.

Kilgard M.J. (Ed). *NVIDIA OpenGL Extension Specifications.* NVIDIA Corporation, 2003. http ://www.nvidia.com/developer.

Kilgariff E. et Fernando R.. *GPU Gems 2 : Programming Techniques for High-Performance Graphics and General-Purpose Computation.* Addison-Wesley, 2005. The GeForce 6 Series GPU Architecture, pp. 471–491.

Kindlmann G. et Durkin J.. *Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering.* In Proceedings of IEEE Symposium on Volume Visualization, 1998. pp. 79–86.

Kipfer P. et Westerman R.. *GPU Construction and Transparent Rendering of Iso-Surfaces.* In Proceedings of IEEE Symposium on Volume Visualization, 2005. pp. 241–248.

Kirihata Y., Leigh J., Xiong C. *et al. A Sort-Last Rendering System Over an Optical Backplane.* In CITSA, 2004.

Kniss J., Kindlmann G. et Hansen C.. *Interactive Volume Rendering Using Multidimensional Transfer Functions and Direct Manipulation Widgets.* In Proceedings of IEEE Visualization Conference, 2001. pp. 255–262.

Kniss J., Kindlmann G. et Hansen C.. *Multidimensional Transfer Functions for Interactive Volume Rendering.* IEEE Transactions on Visualization and Computer Graphics, 2002, vol. 8, n°4, pp. 270–285.

Kniss J., McCormick P., McPherson A. *et al. Interactive Texture-Based Volume Rendering for Large Data Sets.* IEEE Computer Graphics and Applications, 2001, vol. 21, n°4, pp. 52–61.

Kniss J., Premože S., Hansen C. *et al. Interactive Translucent Volume Rendering and Procedural Modeling.* In Proceedings of IEEE Visualization Conference, 2002. pp. 109–116.

Kniss J., Premože S., Hansen C. *et al. A Model for Volume Lighting and Modeling.* IEEE Transactions on Visualization and Computer Graphics, 2003, vol. 9, n°2, pp. 150–162.

Kniss J., Premože S., Ikits M. *et al. Gaussian Transfer Functions for Multi-Field Volume Visualization.* In Proceedings of IEEE Visualization Conference, 2003. pp. 497–504.

Knittel G.. *Using Pre-Integrated Transfer Functions in an Interactive Software System for Volume Rendering.* In Proceedings of Eurographics Short Presentations, 2002. pp. 119–123.

Kraus M.. *Direct Volume Visualization of Geometrically Unpleasant Meshes.* Th. Doct.. University of Stuttgart, 2003.

Kraus M. et Ertl T.. *Adaptive Texture Maps.* In Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware, 2002. pp. 7–15.

Kraus M. et Ertl T.. *The Visualization Handbook.* Elsevier, 2005. Pre-Integrated Volume Rendering, pp. 211–228.

Krüger J. et Westermann R.. *Acceleration Techniques for GPU-Based Volume Rendering.* In Proceedings of IEEE Visualization Conference, 2003. pp. 287–292.

Köse C. et Chalmers A.G.. *Profiling for Efficient Parallel Volume Visualization.* Parallel Computing, 1997, vol. 23, n°7, pp. 943–952.

Labrunye E.. *Extraction Automatique d'Information Géologique à Partir d'Images Sismiques Tridimensionnelles.* Th. Doct.. Institut National Polytechnique de Lorraine, 2004.

Lacroute P.. *Real-Time Volume Rendering on Shared Memory Multiprocessors Using the Shear-Warp Factorization.* In Proceedings of ACM Symposium on Parallel Rendering, 1995. pp. 15–22.

Lacroute P.. *Analysis of a Parallel Volume Rendering System Based on the Shear-Warp Factorization.* IEEE Transactions on Visualization and Computer Graphics, 1996, vol. 2, n°3, pp. 218–231.

Lacroute P. et Levoy M.. *Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transform.* In Proceedings of ACM SIGGRAPH Conference, 1994. pp. 451–457.

LaMar E., Hamann B. et Joy K.I.. *Multiresolution Techniques for Interactive Texture-Based Volume Visualization.* In Proceedings of IEEE Visualization Conference, 1999. pp. 355–361.

Lane D.. *UFAT : A Particle Tracer for Time-Dependent Flow Fields.* In Proceedings of IEEE Visualization Conference, 1994. pp. 257–264.

Larsen B. et Christensen N.. *Real-Time Rendering Using Smooth Hardware Optimized Level of Detail.* In Proceedings of WSCG, 2003.

Laudon J. et Lenoski D.. *The SGI Origin : A ccNUMA Highly Scalable Server.* In Proceedings of the 24^{th} Annual International Symposium on Computer Architecture, 1997. pp. 241–251.

Law A. et Yagel R.. *Multi-Frame Thrashless Ray Casting with Advancing Ray-Front.* In Proceedings of Graphics Interface, 1996.

Lee T.Y., Raghavendra C.S. et Nicholas J.B.. *Image Composition Schemes for Sort-Last Polygon Rendering on 2D Mesh Multicomputers.* IEEE Transactions on Visualization and Computer Graphics, 1996, vol. 2, n°3, pp. 202–217.

Levenberg J.. *Fast View-Depdendent Level-of-Detail Rendering Using Cached Geometry.* In Proceedings of IEEE Visualization Conference, 2002. pp. 259–266.

Levoy M.. *Display of Surfaces from Volume Data.* IEEE Computer Graphics and Applications, 1988, vol. 8, n°5, pp. 29–37.

Levoy M.. *Efficient Ray Tracing of Volume Data.* ACM Transactions on Graphics, 1990, vol. 9, n°3, pp. 245–261.

Li K.. *Shared Virtual Memory on Loosely Coupled Multiprocessors.* Th. Doct.. Department of Computer Science, Yale University, 1986.

Li P., Whitman S., Mendoza R. *et al. ParVox - A Parallel Splatting Volume Rendering System for Distributed Visualization.* In Proceedings of ACM Symposium on Parallel Rendering, 1997. pp. 7–14.

Li W. et Kaufman A.. *Texture Partitioning and Packing for Accelerating Texture-Based Volume Rendering.* In Proceedings of Graphics Interface, 2003. pp. 81–88.

Li W., Mueller K. et Kaufman A.. *Empty Space Skipping and Occlusion Clipping for Texture-Based Volume Rendering.* In Proceedings of IEEE Visualization Conference, 2003. pp. 317–324.

Lichtenbelt B., Crane R. et Navqi S.. *Introduction to Volume Rendering.* Prentice Hall, 1998. 236 p.

Lin C.F., Yang D.L. et Chung Y.C.. *Parallel Shear-Warp Factorization Volume Rendering Using Efficient 1-D and 2-D Partitioning Schemes for Distributed Memory Multicomputers.* Journal of Supercomputing, 2002, vol. 22, n°3, pp. 277–302.

Lindstrom P., Koller D., Ribarsky W. *et al. Real-Time, Continuous Level of Detail Rendering of Height Fields.* In Proceedings of ACM SIGGRAPH Conference, 1996. pp. 109–118.

Lindstrom P. et Pascucci V.. *Terrain Simplification Simplified : A General Framework for View-Dependent Out-of-Core Visualization.* IEEE Transactions on Visualization and Computer Graphics, 2002, vol. 8, n°3, pp. 239–254.

Lombeyda S., Moll L., Shand M. *et al. Scalable Interactive Volume Rendering Using Off-the-Shelf Components.* In Proceedings of IEEE Symposium on Parallel and Large-Data Visualization and Graphics, 2001. pp. 115–121.

Lorensen L. et Cline H.. *Marching Cubes : A High Resolution 3D Surface Construction Algorithm.* In Proceedings of ACM SIGGRAPH Conference, 1987. pp. 163–169.

Losasso F. et Hoppe H.. *Geometry Clipmaps : Terrain Rendering Using Nested Regular Grids.* In Proceedings of ACM SIGGRAPH Conference, 2004. pp. 769–776.

Luebke D., Reddy M., Cohen J. *et al. Level of Detail for 3D Graphics.* Morgan Kaufman, 2003. 390 p.

Lum E.B., Wilson B. et Ma K.L.. *High-Quality Lighting and Efficient Pre-Integration for Volume Rendering.* In Proceedings of Eurographics/IEEE Symposium on Visualization, 2004. pp. 25–34.

Lévy B., Caumon G., Conreaux S. *et al. Circular Incident Edges List : A Data Structure for Rendering Complex Unstructured Grids.* In Proceedings of IEEE Visualization Conference, 2001.

Ma K.L., Painter J.S., Hansen C.D. *et al. Parallel Volume Rendering Using Binary-Swap Image Composition.* IEEE Computer Graphics and Applications, 1994, vol. 14, n°4, pp. 59–68.

Machiraju R.K. et Yagel R.. *Efficient Feed-Forward Volume Rendering Techniques for Vector and Parallel Processors.* In Proceedings of Supercomputing Conference, 1993. pp. 699–708.

Mallet J.L.. *Discrete Smooth Interpolation in Geometric Modeling.* ACM Transactions on Graphics, 1989, vol. 8, n°2, pp. 121–144.

Mallet J.L.. *Geomodeling.* Oxford University Press, 2002. Applied Geostatistics Series. 608 p.

Mark W.R., Glanville R.S., Akeley K. *et al. Cg : A System for Programming Graphics Hardware in a C-like Language.* In Proceedings of ACM SIGGRAPH Conference, 2003. pp. 896–907.

Marsh A.J.. *Volume Roaming and Volume Visualization of Large 3D Datasets.* In Proceedings of the Indonesian Petroleum Association 28^{th} Annual Meeting, 2001. pp. 195–209.

Marsh A.J., Kidd G.D. et Furniss A.. *3D Seismic Visualization Using Multiple Volume Data Sets.* In Proceedings of the Indonesian Petroleum Association 27^{th} Annual Meeting, 2000. pp. 599–612.

Max N.. *Optical Models for Direct Volume Rendering.* IEEE Transactions on Visualization and Computer Graphics, 1995, vol. 1, n°2, pp. 99–108.

Max N., Hanrahan P. et Crawfis R.. *Area and Volume Coherence for Efficient Visualization of 3D Scalar Functions.* In ACM Compter Graphics (San Diego Workshop on Volume Visualization), 1990. pp. 27–33.

McCormick B.H., DeFanti T.A. et Brown M.D. (Eds). *Visualization in Scientific Computing.* U.S. National Science Foundation, 1987.

McCreight E.M.. *Priority Search Trees.* SIAM Journal of Computing, 1985, vol. 14, n°2, pp. 257–276.

Meissner M., Guthe S. et Strasser W.. *Interactive Lighting Models and Pre-Integration for Volume Rendering on PC Graphics Accelerators.* In Proceedings of Graphics Interface, 2002. pp. 209–218.

Meissner M., Hoffmann U. et Strasser W.. *Enabing Classification and Shading for 3D Texture Mapping Based Volume Rendering Using OpenGL and Extensions.* In Proceedings of IEEE Visualization Conference, 1999. pp. 207–214.

Meissner M., Kanus U. et Strasser W.. *VIZARD-II : A PCI-Card for Real-Time Volume Rendering.* In Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware, 1998. pp. 61–67.

Meissner M., Kanus U., Wetekam G. *et al. VIZARD-II : A Reconfigurable Interactive Volume Rendering System.* In Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware, 2002. pp. 137–146.

Microsoft Corporation. *DirectX 9.0 Graphics.* http ://msdn.microsoft.com/directx, 2002.

Microsoft Corporation. *High-Level Shader Language.* DirectX 9.0 Graphics, http ://msdn.microsoft.com/directx, 2002.

Möller T. et Haines E.. *Real-Time Rendering, Second Edition.* A K Peters, 2002. 835 p.

Möller T., Machiraju R., Mueller K. *et al. Evaluation and Design of Filters Using a Taylor Series Expansion.* IEEE Transactions on Visualization and Computer Graphics, 1999, vol. 3, n°2, pp. 184–199.

Molnar S., Cox M., Ellsworth D. *et al. A Sorting Classification of Parallel Rendering.* IEEE Computer Graphics and Applications, 1994, vol. 14, n°4, pp. 23–32.

Montani C., Perego R. et Scopigno R.. *Parallel Volume Visualization on a Hypercube Architecture.* In Proceedings of IEEE Symposium on Volume Visualization, 1992. pp. 9–16.

Moore D. et Warren J.. *Mesh Displacement : An Improved Contouring Method for Trivariate Data*. Technical Report TR-91-166, Rice University, 1991.

Mueller K. et Crawfis R.. *Eliminating Popping Artifacts in Sheet-Buffer Based Splatting*. In Proceedings of IEEE Visualization Conference, 1998. pp. 239–245.

Mueller K., Möller T. et Crawfis R.. *Splatting Without the Blur*. In Proceedings of IEEE Visualization Conference, 1999. pp. 363–370.

Neumann U.. *Parallel Volume Rendering Algorithm Performance on Mesh-Connected Multicomputers*. In Proceedings of ACM Symposium on Parallel Rendering, 1993.

Nguyen K.G. et Saupe D.. *Rapid High Quality Compression of Volume Data for Visualization*. In Proceedings of Eurographics/IEEE Symposium on Visualization, 2001.

Nieh J. et Levoy M.. *Volume Rendering on Scalable Shared Memory MIMD Architectures*. In Workshop on Volume Visualization, 1992. pp. 17–24.

Nielson G.M. et Hamann B.. *The Asymptotic Decider : Resolving the Ambiguity in Marching Cubes*. In Proceedings of IEEE Visualization Conference, 1991. pp. 83–91.

Nulkar M. et Mueller K.. *Splatting with Shadows*. In Proceeding of International Workshop on Volume Graphics, 2001.

NVIDIA Corporation. *Fast Texture Downloads and Readbacks Using Pixel Buffer Objects in OpenGL*. http ://developer.nvidia.com/object/fast_texture_transfers.html, 2005.

O'Brien M.J. et Gray S.H.. *Can We Image Beneath Salt ?* The Leading Edge, 1996, vol. 15, n°1, pp. 17–22.

Osborne R., Pfister H., Lauer H. *et al. EM-Cube : An Architecture for Low-Cost Real-Time Volume Rendering*. In Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware, 1997. pp. 131–138.

Owens J.D.. *GPU Gems 2 : Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley, 2005. Streaming Architectures and Technology Trends, pp. 457–470.

Owens J.D., Luebke D., Govindaraju N. *et al. A Survey of General-Purpose Computation on Graphics Hardware*. In Proceedings of Eurographics/IEEE Symposium on Visualization, 2005. pp. 21–51.

Pajarola R.B.. *Large Scale Terrain Visualization Using the Restricted Quadtree Triangulation*. In Proceedings of IEEE Visualization Conference, 1998. pp. 19–26.

Parker S., Parker M., Livnat Y. *et al. Interactive Ray Tracing for Volume Visualization*. IEEE Transactions on Visualization and Computer Graphics, 1999, vol. 5, n°3, pp. 238–250.

Pascucci V.. *Isosurface Computation Made Simple : Hardware Acceleration, Adaptive Refinement and Tetrahedral Stripping*. In Proceedings of Eurographics/IEEE Symposium on Visualization, 2004. pp. 293–300.

Pfister H.. *The Visualization Handbook*. Elsevier, 2005. Hardware-Accelerated Volume Rendering, pp. 229–258.

Pfister H., Hardenbergh J., Knittel J. *et al. The VolumePro Real-Time Ray Casting System*. In Proceedings of ACM SIGGRAPH Conference, 1999. pp. 251–260.

Pfister H. et Kaufman A.. *Cube-4 : A Scalable Architecture for Real-Time Volume Rendering*. In Proceedings of IEEE Symposium on Volume Visualization, 1996. pp. 47–54.

Pfister H., Kaufman A. et Chiueh T.. *Cube-3 : A Real-Time Architecture for High-Resolution Volume Visualization.* In Proceedings of IEEE Symposium on Volume Visualization, 1994. pp. 75–82.

Pfister H., Lorensen W., Bajaj C. *et al. The Transfer Function Bake-Off.* IEEE Computer Graphics and Applications, 2001, pp. 16–22.

Pharr M. (Ed). *GPU Gems 2 : Programming Techniques for High-Performance Graphics and General-Purpose Computation.* Addison-Wesley, 2005. 814 p.

Phong B.T.. *Illumination for Computer Generated Pictures.* Communications of the ACM, 1975, vol. 18, n°6, pp. 311–317.

Plate J., Tirtasana M., Carmona R. *et al. Octreemizer : A Hierarchical Approach for Interactive Roaming Through Very Large Volumes.* In Proceedings of Eurographics/IEEE Symposium on Visualization, 2002.

Porter D.H.. *Volume Visualization of High Resolution Data Using PC-Clusters.* Technical Report, University of Minnesota, 2002.

Porter T. et Duff T.. *Compositing Digital Images.* In Proceedings of ACM SIGGRAPH Conference, 1984. pp. 253–259.

Ray H., Pfister H., Silver D. *et al. Ray-Casting Architectures for Volume Visualization.* IEEE Transactions on Visualization and Computer Graphics, 1999, vol. 5, n°3, pp. 210–223.

Rezk-Salama C.. *Volume Rendering Techniques for General Purpose Graphics Hardware.* Th. Doct.. University of Erlangen-Nüremberg, 2001.

Rezk-Salama C., Engel K., Bauer M. *et al. Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization.* In Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware, 2000. pp. 109–118.

Rezk-Salama C., Engel K., Hadwiger M. *et al. Real-Time Volume Graphics.* Course 28, ACM SIGGRAPH, 2004.

Robein E.. *Vitesses et Techniques d'Imagerie en Sismique Réflexion - Principes et Méthodes.* Paris : Tec & Doc, 1999. 387 p.

Rost R.. *OpenGL Shading Language.* Addison-Wesley, 2004. 565 p.

Röttger S. et Ertl T.. *A Two-Step Approach for Interative Pre-Integrated Volume Rendering of Unstructured Grids.* In Proceedings of IEEE Symposium on Volume Visualization, 2002. pp. 23–28.

Röttger S., Guthe S., Weiskopf D. *et al. Smart Hardware-Accelerated Volume Rendering.* In Proceedings of Eurographics/IEEE Symposium on Visualization, 2003. pp. 231–238.

Röttger S., Heidrich W., Slussallek P. *et al. Real-Time Generation of Continuous Levels of Detail for Height Fields.* In Proceedings of WSCG, 1998. pp. 315–322.

Röttger S., Kraus M. et Ertl T.. *Hardware-Accelerated Volume and Isosurface Rendering Based on Cell Projection.* In Proceedings of IEEE Visualization Conference, 2000. pp. 109–116.

Sabella P.. *A Rendering Algorithm for Visualizing 3D Scalar Fields.* In Proceedings of ACM SIGGRAPH Conference, 1988. pp. 51–58.

Samet H.. *The Quadtree and Related Hierarchical Data Structures.* ACM Computing Surveys, 1984, vol. 16, n°2, pp. 187–260.

Samet H.. *Applications of Spatial Data Structures.* Addison Wesley, 1990.

Sano K., Kitajima H., Kobayashi H. *et al. Parallel Processing of the Shear-Warp Factorization with the Binary-Swap Method on a Distributed Memory Multiprocessor.* In Proceedings of ACM Symposium on Parallel Rendering, 1997.

Sano K., Kobayashi Y. et Nakamura T.. *Differential Coding Scheme for Efficient Parallel Image Composition on a PC Cluster System.* Parallel Computing, 2004, vol. 30, n°2, pp. 285–299.

Schneider J. et Westermann R.. *Compression Domain Volume Rendering.* In Proceedings of IEEE Visualization Conference, 2003. pp. 293–300.

Schroder F. et Rossbach P.. *Managing the Complexity of Digital Terrain Models.* Computer & Graphics, 1994, vol. 18, n°6, pp. 775–783.

Schroeder W., Zarge J. et Lorensen W.. *Decimation of Triangle Meshes.* Proceedings of ACM SIGGRAPH Conference, 1992, pp. 65–70.

Schulze J.P., Kraus M., Lang U. *et al. Integrating Pre-Integration into the Shear-Warp Algorithm.* In Proceedings of Eurographics/IEEE Symposium on Visualization, 2003. pp. 109–118.

Segal M. et Akeley K.. *The OpenGL Graphics System : A Specification (Version 1.5).* Silicon Graphics, 2003. http ://www.opengl.org.

Selley R.C.. *Elements of Petroleum Geology, Second Edition.* San Diego : Academic Press, 1998. 470 p.

Sheriff R.E.. *Inferring Stratigraphy from Seismic Data.* AAPG Bulletin, 1976, vol. 60, pp. 528–542.

Silberschatz A., Gagne G. et Galvin P.B.. *Operating System Concepts, Seventh Edition.* Hoboken : John Wiley & Sons, Inc., 2005. 921 p.

Silicon Graphics Inc.. *SGI Origin 3000 Datasheet.* http ://www.sgi.com/origin/3000/datasheet.htm, 2002.

Silva C.T., Kaufman A.E. et Pavlakos C.. *PVR : High-Performance Volume Rendering.* IEEE Computational Science and Engineering, 1996, vol. 3, n°4, pp. 18–28.

Silva P.M., Machado M. et Gattass M.. *3D Seismic Volume Rendering.* In Brazilian Geophysical Society (SBGf) 8^{th} International Congress, 2003.

SiSoftware Sandra 2007. http ://www.sisoftware.co.uk/.

Smith A.R.. *A Pixel Is Not A Little Square, A Pixel Is Not A Little Square, A Pixel Is Not A Little Square ! (And a Voxel is Not a Little Cube).* Technical Report Microsoft, Inc., 1995.

Sollenberg R.L. et Milgram P.. *Effects of Stereoscopic and Rotational Displays in a 3D Path Tracing Task.* In Human Factors, 1993. pp. 483–499.

Stein C., Becker B. et Max N.. *Sorting and Hardware Assisted Rendering for Volume Visualization.* In Proceedings of IEEE Symposium on Volume Visualization, 1994. pp. 83–89.

Stollnitz E.J., DeRose T.D. et Salesin D.H.. *Wavelets for Computer Graphics : A Primer, Part 1.* IEEE Computer Graphics and Applications, 1995, vol. 15, n°3, pp. 76–84.

Stollnitz E.J., DeRose T.D. et Salesin D.H.. *Wavelets for Computer Graphics : A Primer, Part 2.* IEEE Computer Graphics and Applications, 1995, vol. 15, n°4, pp. 75–85.

Stompel A., Ma K.L., Lum E.B. *et al. SLIC : Scheduled Linear Image Composition for Parallel Volume Rendering.* In Proceedings of IEEE Symposium on Parallel and Large-Data Visualization and Graphics, 2003. pp. 33–40.

Strengert M., Magallon M., Weiskopf D. *et al. Hierarchical Visualization and Compression of Large Volume Datasets Using GPU Clusters.* In Proceedings of Eurographics Symposium on Parallel Graphics and Visualization, 2004.

Takeuchi A., Ino F. et Hagihara K.. *An Improved Binary-Swap Compositing for Sort-Last Parallel Rendering on Distributed Memory Multiprocessors.* Parallel Computing, 2003, vol. 29, n°6, pp. 11–12.

Tanenbaum A.S.. *Modern Operating Systems, Second Edition.* Prentice Hall, 2001. 976 p.

Taner M.T.. *Seismic Attributes.* Recorder, 2001, vol. 26, n°9, pp. 48–56.

Taner M.T. et Koehler. *Velocity Spectra - Digital Computer Derivation and Application of Velocity Functions.* Geophysics, 1969, vol. 34, n°6, pp. 859–881.

Taner M.T., Koehler F. et Sheriff R.E.. *Complex Seismic Trace Analysis.* Geophysics, 1979, vol. 44, n°6, pp. 1041–1063.

Taner M.T. et Sheriff R.E.. *Application of Amplitude, Frequency, and Other Attributes to Stratigraphic and Hydrocarbon Determination.* Memories - AAPG, 1977, vol. 26, pp. 301–327.

Taylor C. et Lima J.. *How Onboard 3D Visualization Technology Benefits Marine Seismic Acquisition.* First Break, 2005, vol. 23, pp. 65–66.

Thakur R. et Gropp W.. *Improving the Performance of Collective Operations in MPICH.* In Proceedings of the 10^{th} European PVM/MPI Users' Group Conference (Euro PVM/MPI 2003), 2003. pp. 257–267.

Tuy H. et Tuy L.. *Direct 2D Display of 3D Objects*, 1984, vol. 4, n°10, pp. 20–33.

Ueng S.K., Siborski K. et Ma K.L.. *Out-of-Core Streamline Visualization on Large Unstructured Meshes.* ICASE Report 97-22, NASA Langley Research Center, 1997.

Vail P.R., Mitchum R.M., Todd R.G. *et al. Seismic Stratigraphy and Global Changes of Sea Level.* Memories - AAPG, 1977, vol. 26, pp. 49–212.

Wagner D.. *ShaderX^2 : Shader Programming Tips and Tricks.* Wordware, 2004. Terrain Geomorphing in the Vertex Shader.

Wald I., Dietrich A. et Slusallek P.. *An Interactive Out-of-Core Rendering Framework for Visualizing Massively Complex Models.* In Proceedings of Eurographics Symposium on Rendering, 2004. pp. 81–92.

Weiler M., Kraus M., Merz M. *et al. Hardware-Based View-Independent Cell Projection.* IEEE Transactions on Visualization and Computer Graphics, 2003, vol. 9, n°2, pp. 163–175.

Weiler M., Westermann R., Hansen C. *et al. Level-Of-Detail Volume Rendering via 3D Textures.* In Proceedings of IEEE Symposium on Volume Visualization, 2000. pp. 7–13.

Weiskopf D., Engel K. et Ertl T.. *Interactive Clipping Techniques for Texture-Based Volume Visualization and Volume Shading.* IEEE Transactions on Visualization and Computer Graphics, 2003, vol. 9, n°3, pp. 298–312.

Weisstein E.. *Wolfram MathWorld.* http ://www.mathworld.wolfram.com.

Westermann R. et Ertl T.. *Efficiently Using Graphics Hardware in Volume Rendering Applications.* In Proceedings of ACM SIGGRAPH Conference, 1998. pp. 169–177.

Westover L.. *Interactive Volume Rendering.* In Proceedings of Chapell Hill Volume Visualization Workshop, 1989. pp. 9–16.

Westover L.. *Footprint Evaluation for Volume Rendering.* In Proceedings of ACM SIGGRAPH Conference, 1990. pp. 367–376.

WHITMAN S.. *A Task Adaptive Parallel Graphics Renderer.* In Proceedings of ACM Symposium on Parallel Rendering, 1993. pp. 27–34.

WILHELMS J. et VAN GELDER A.. *Octrees for Faster Isosurface Generation.* ACM Transactions on Graphics, 1992, vol. 11, n°3, pp. 201–227.

WILLIAMS P., MAX N. et STEIN C.. *A High Accuracy Volume Renderer for Unstructured Data.* IEEE Transactions on Visualization and Computer Graphics, 1998, vol. 4, n°1, pp. 37–54.

WITTENBRINK C.M., MALZBENDER T. et GOSS M.E.. *Opacity-Weighted Color Interpolation Volume Sampling.* In Proceedings of IEEE Symposium on Volume Visualization, 1998. pp. 135–142.

WOO M., NEIDER J., DAVIS T. *et al. OpenGL Programming Guide : The Official Guide to Learning OpenGL, Version 1.2, Third Edition.* Addison-Wesley, 1999.

WU Y., BHATIA V., LAUER H. *et al. Shear-Image Ray-Casting Volume Rendering.* In Proceedings of ACM Symposium on Interactive 3D Graphics, 2003. pp. 152–162.

YANG D.L., YU J.C. et CHUNG Y.C.. *Efficient Compositing Methods for the Sort-Last-Sparse Parallel Volume Rendering System on Distributed Memory Multicomputers.* Journal of Supercomputing, 2001, vol. 18, n°2, pp. 201–220.

YILMAZ O.. *Seismic Data Processing (SEG Investigations in Geophysics 2).* Tulsa : SEG, 1987.

ZARANTONELLO S.E. AND BEVC D.. *Compression of Seimsic Data Using Ridgelets.* In Society of Exploration Geophysicists (SEG) 75^{th} Annual Meeting, 2005.

ZEKAUSKAS M.J., SAWDON W.A. et BERSHAD B.N.. *Software Write Detection for Distributed Shared Memory.* In Proceedings of the Symposium on Operating System Design and Implementation, 1994. pp. 87–100.

ZHANG C. et CRAWFIS R.. *Volumetric Shadows Using Splatting.* In Proceedings of IEEE Visualization Conference, 2002. pp. 85–92.

U.S. NATIONAL LIBRARY OF MEDICINE, National Institute of Health. *The Visible Human Project.* http ://www.nlm.nih.gov/research/visible/visible_human.html.

Résumé

Visualisation de Données Volumiques Massives - Applications aux Données Sismiques

Les données de sismique réflexion sont une source d'information essentielle pour la modélisation tridimensionnelle des structures du sous-sol dans l'exploration-production des hydrocarbures. Ce travail vise à fournir des outils de visualisation pour leur interprétation. Les défis à relever sont à la fois d'ordre qualitatif et quantitatif. Il s'agit en effet de considérer (1) la nature particulière des données et la démarche d'interprétation (2) la taille des données. Notre travail s'est donc axé sur ces deux aspects :

1. Du point de vue qualitatif, nous mettons tout d'abord en évidence les principales caractéristiques des données sismiques, ce qui nous permet d'implanter une technique de visualisation volumique adaptée. Nous abordons ensuite l'aspect multimodal de l'interprétation qui consiste à combiner plusieurs sources d'information (sismique et structurale). Selon la nature de ces sources (strictement volumique ou volumique et surfacique), nous proposons deux systèmes de visualisation différents.
2. Du point de vue quantitatif, nous définissons tout d'abord les principales contraintes matérielles intervenant dans l'interprétation, ce qui nous permet d'implanter un système générique de gestion de la mémoire. Initialement destiné au couplage de la visualisation et des calculs sur des données volumiques massives, il est ensuite amélioré et spécialisé pour aboutir à un système dynamique de gestion distribuée de la mémoire sur cluster de PCs. Cette dernière version, dédiée à la visualisation, permet de manipuler des données sismiques à échelle régionale (100-200 Go) en temps réel.

Les problématiques sont abordées à la fois dans le contexte scientifique de la visualisation et dans le contexte d'application des géosciences et de l'interprétation sismique.

Mots-clés: Interprétation sismique, visualisation volumique, visualisation multimodale, matériel graphique programmable, données volumiques massives, gestion de la mémoire, clusters graphiques.

Abstract

Visualization of Massive Data Volumes - Applications to Seismic Data

Seismic reflection data are a valuable source of information for the three-dimensional modeling of subsurface structures in the exploration-production of hydrocarbons. This work focuses on the implementation of visualization techniques for their interpretation. We face both qualitative and quantitative challenges. It is indeed necessary to consider (1) the particular nature of seismic data and the interpretation process (2) the size of data. Our work focuses on these two distinct aspects :

1. From the qualitative point of view, we first highlight the main characteristics of seismic data. Based on this analysis, we implement a volume visualization technique adapted to the specificity of the data. We then focus on the multimodal aspect of interpretation which consists in combining several sources of information (seismic and structural). Depending on the nature of these sources (strictly volumes or both volumes and surfaces), we propose two different visualization systems.
2. From the quantitative point of view, we first define the main hardware constraints involved in seismic interpretation. Focused on these constraints, we implement a generic memory management system. Initially able to couple visualization and data processing on massive data volumes, it is then improved and specialised to build a dynamic system for distributed memory management on PC clusters. This later version, dedicated to visualization, allows to manipulate regional scale seismic data (100-200 GB) in real-time.

The main aspects of this work are both studied in the scientific context of visualization and in the application context of geosciences and seismic interpretation.

Keywords: Seismic interpretation, volume visualization, multimodal visualization, programmable graphics hardware, massive data volumes, memory management, graphics clusters.

Centre de Recherches Pétrographiques et Géochimiques
Laboratoire d'Infographie et d'Analyse de Données
Rue du Doyen Marcel Roubault - 54500 Vandœuvre

INSTITUT NATIONAL
POLYTECHNIQUE
DE LORRAINE

AUTORISATION DE SOUTENANCE DE THESE
DU DOCTORAT DE L'INSTITUT NATIONAL
POLYTECHNIQUE DE LORRAINE

o0o

VU LES RAPPORTS ETABLIS PAR :
Monsieur Charles D. HANSEN, Professeur, Université de l'Utah, USA

Monsieur Helmut SCHAEBEN, Professeur, Technische Universität Bergakademie Freiberg, Germany

Le Président de l'Institut National Polytechnique de Lorraine, autorise :

Monsieur CASTANIÉ Laurent

NANCY BRABOIS
2, AVENUE DE LA
FORET-DE-HAYE
BOITE POSTALE 3
F - 54501
VANDŒUVRE CEDEX

à soutenir devant un jury de l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE, une thèse intitulée :

"Visualisation de données volumiques massives - Applications aux données sismiques"

en vue de l'obtention du titre de :

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : « **Géosciences** »

Fait à Vandoeuvre, le 06 novembre 2006
Le Président de l'I.N.P.L.,
L. SCHUFFENECKER

TEL. 33/03.83.59.59.59
FAX. 33/03.83.59.59.55