JEAN-DANIEL DESCHÊNES

Modélisation interactive : amélioration du processus de reconstruction d'un modèle 3D par la compression temps réel.

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en génie électrique
pour l'obtention du grade de Maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

2008

Résumé

Ces travaux présentent un système interactif de modélisation 3D multirésolution permettant la compression en temps réel de la surface à reconstruire. Les principaux avantages de ce système par rapport à ceux présentés dans le passé sont de pouvoir localement reconstruire la surfacè à différents niveaux de résolution et de compresser la surface durant l'acquisition des données brutes. Cette utilisation judicieuse de la mémoire rend désormais possible la modélisation d'objets de plus grande taille ou à une résolution plus élevée. Le document est divisé en trois parties.

Tout d'abord, nous ferons un retour sur la représentation de surface qui est à la base du système proposé : le champ vectoriel. Nous montrerons tous les avantages d'une telle représentation dans le contexte de la modélisation interactive.

Par après, nous aborderons le développement de la représentation multirésolution s'inspirant du champ vectoriel et permettant la compression en temps réel. Nous verrons comment il est possible de faire cohabiter différents niveaux de résolution à l'intérieur d'une même structure de données tout en conservant une représentation cohérente de la surface. Ensuite, nous expliquerons tous les algorithmes nécessaires à la compression en temps réel.

La dernière partie de ces travaux aborde la mise au point d'un module de visualisation permettant d'afficher l'état de la surface multirésolution durant l'acquisition des données brutes. L'approche utilisée repose sur une technique de *lancer de rayons* et offre une grande qualité de rendu tout en conservant l'interactivité du système.

Abstract

This work presents a multiresolution interactive 3D modeling system that allows real time compression of the surface being reconstructed. The main benefits of this system compared to those presented in the past are the possibility to locally reconstruct the surface at different resolution levels and to be able to compress the surface during the acquisition of raw data. This optimization of the memory makes it possible to model objects of bigger size or at a higher resolution. The document is divided into three parts.

First of all, we will revisit the representation of surface used by the proposed system : the vector field. We will show all the benefits of such a representation in the context of interactive modeling.

Then, we will approach the development of the multiresolution representation taking as a starting point the vector field and allowing real time compression of the surface. We will see how it is possible to reconstruct different levels of resolution inside the same data structure while preserving a coherent representation of the surface. Then, we will explain all the algorithms required by the real time compression.

The last part of this work presents a visualization module that allows the render of the surface while acquiring raw data. The visualization process is based on a *ray caster* and offers a great rendering quality while preserving the interactivity of the system.

Avant-propos

Tout au long de ce document, j'utiliserai l'expressions *octree* qui peut sembler être un anglicisme. Cependant, il n'en est rien, car le mot *octree* est accepté par *l'Office Québécoise de la Langue Française* via le *Grand Dictionnaire Terminologique* (www.granddictionnaire.com) et représente effectivement une version tridimensionnelle d'un arbre binaire classique.

Pour ce qui est de l'expression *voxel*, je n'ai pas trouvé de contrepartie française. J'espère que vous m'excuserez de son utilisation.

Table des matières

Résumé ii

Abstract iii

Avant-propos iv

Table des matières v

1 Introduction 1
- 1.1 Modélisation 3D . . . 1
- 1.2 Modélisation 3D interactive . . . 2
- 1.3 Mise en contexte . . . 4
- 1.4 Définition du problème et objectifs . . . 7
  - 1.4.1 Objectif 1 : Amélioration de la gestion de la mémoire . . . 8
  - 1.4.2 Objectif 2 : Amélioration du module de visualisation . . . 10
- 1.5 Plan du mémoire . . . 12

2 Le champ vectoriel pour la reconstruction de surfaces 13
- 2.1 Introduction . . . 13
- 2.2 Définition d'un champ vectoriel . . . 14
- 2.3 Enveloppe de reconstruction et cellule fondamentale . . . 15
- 2.4 Retour sur la boucle de modélisation interactive . . . 17
  - 2.4.1 Acquisition des données brutes . . . 17
  - 2.4.2 Alignement et fusion des données . . . 21
  - 2.4.3 Reconstruction de la surface . . . 23
  - 2.4.4 Visualisation . . . 31
- 2.5 Implantation monorésolution . . . 35
- 2.6 Conclusion . . . 38

3 Représentation multirésolution dans un champ vectoriel 39
- 3.1 Introduction . . . 39
- 3.2 Avantages de la multirésolution . . . 40
- 3.3 Champ vectoriel multirésolution . . . 42

3.4 Interprétation et cohérence du champ vectoriel 45
3.4.1 Interprétation du champ vectoriel 45
3.4.2 Cohérence du champ vectoriel 46
3.5 Expérimentations 53
3.5.1 Implications pratiques 53
3.5.2 Densité VS résolution 54
3.6 Algorithme de reconstruction multirésolution 56

4 Sélection automatique du niveau de résolution 62
4.1 Introduction 62
4.2 Reconstruction progressive des niveaux de résolution grossiers vers fins 63
4.2.1 Stabilité de la surface 64
4.2.2 Algorithme de reconstruction progressive 67
4.2.3 Surcharge due à la reconstruction progressive 68
4.2.4 Facteurs influençant la densité de points pour atteindre la stabilité 70
4.3 Compression temps réel des niveaux fins vers grossiers 72
4.3.1 Condition d'ajustement 73
4.3.2 Algorithme de compression temps réel 76
4.4 Compression hors ligne 79
4.5 Expérimentations 80
4.5.1 Scénario d'acquisition typique 80
4.5.2 Gestion de la mémoire 83
4.5.3 Reconstruction à haute résolution 88
4.5.4 Reconstruction à partir de données réelles 92
4.5.5 Limitations actuelles 92
4.6 Conclusion 94

5 Visualisation de la surface 95
5.1 Introduction 95
5.2 Approche par lancer de rayons 97
5.2.1 Stratégie de détection de la surface 99
5.2.2 Stratégie de parcours du volume 105
5.3 Approche fovéale 115
5.4 Conclusion 120

6 Conclusion 121

A Implantation de la traversée paramétrique de l'*octree* 126

Bibliographie 129

Chapitre 1

Introduction

1.1 Modélisation 3D

De nos jours, les modèles numériques d'objets nous côtoient constamment. Communément appelées *modèles 3D*, ces représentations numériques de la réalité sont utilisées de diverses façons. Les films les utilisent pour pousser davantage leurs effets spéciaux et rendre les cascades encore plus époustouflantes. Prenons l'exemple du film *The Matrix* [37] qui a utilisé les services de la compagnie canadienne *XYZ-RGB Imaging Services* [38] pour modéliser la figure des acteurs dans le but de générer des scènes de combat à couper le souffle. Les jeux vidéo en font usage dans le but d'améliorer le réalisme des environnements dans lesquels les joueurs sont plongés. Les concepteurs de la dernière version du jeu *Formula One Championship Edition*, au Playstation 3, a également fait appel à *XYZ-RGB* pour modéliser la figure de chaque pilote dans le but d'atteindre un réalisme inégalé [39]. Le domaine de la publicité n'échappe pas à cette vague du numérique. Il est désormais fréquent de voir de simples publicités utiliser le numérique pour rendre leurs messages encore plus percutants. Grâce aux avancements scientifiques récemment réalisés dans le domaine de la modélisation 3D par des compagnies canadiennes [36, 35, 38], cette tendance à l'utilisation du numérique augmentera, car il est de plus en plus facile de modéliser en 3D les objets qui nous entourent.

De manière spécifique, la modélisation 3D de la géométrie d'un objet est un domaine de la vision numérique visant à générer un modèle numérique d'un objet à partir de mesures. Généralement, les mesures 3D sont obtenues à l'aide d'un laser balayant un objet observé par une ou plusieurs caméras. D'autres systèmes reposent simplement sur l'observation passive d'un objet sous différents points de vue, également avec une ou plusieurs caméras. Une fois l'objet numérisé, il est possible d'en utiliser le modèle

dans différentes situations comme dans un film, un jeu vidéo ou pour reconstruire un environnement virtuel quelconque.

Les premiers travaux dans le domaine de la modélisation 3D de la géométrie d'objets ont été réalisés il y a plus de 25 ans. Aujourd'hui, la disponibilité d'ordinateurs de table de plus en plus puissants favorise grandement le développement de systèmes complexes et performants permettant ainsi d'atteindre une numérisation de plus grande qualité en un temps beaucoup plus court.

Ces dernières années, le principal défi dans ce champ de recherche n'est plus simplement l'obtention d'un modèle numérique de haute qualité d'un objet. Dorénavant, le modèle de l'objet doit être obtenu rapidement [31, 25]. Il n'est plus question d'attendre des heures que les données soient traitées dans un laboratoire universitaire. Un utilisateur doit désormais être en mesure avec un système peu encombrant, basé sur un simple portable et un capteur tenu en main, de numériser des objets tout en visualisant le résultat durant l'acquisition des données brutes [7].

Cette nouvelle ligne de pensée, appelée modélisation 3D interactive, est alimentée par la demande croissante de l'industrie pour des systèmes permettant la modélisation d'objets en temps réel. L'utilisateur doit être en mesure de voir le résultat de la modélisation durant l'acquisition pour vérifier la couverture complète de l'objet et, plus important encore, valider la qualité du modèle en cours de reconstruction. À la fin de l'acquisition, le modèle 3D doit être prêt à être utilisé sans réclamer de longs traitements supplémentaires.

1.2 Modélisation 3D interactive

Le but premier de la modélisation interactive est de permettre à l'utilisateur de voir le résultat de la modélisation en temps réel pour valider la qualité du modèle 3D obtenu ou utiliser le modèle 3D dans un futur rapproché. Différents domaines d'application bénéficient grandement de cette nouvelle flexibilité apportée par la modélisation interactive. En voici quelques exemples.

Tout d'abord, un tel système peut être utilisé en inspection industrielle au niveau de l'évaluation de la qualité de fabrication des pièces. Dans une industrie machinant des pièces de précision, un certain nombre d'échantillons peuvent être modélisés par un employé et le modèle 3D obtenu comparé à un modèle 3D de référence et ce, en temps réel. Ainsi, si les pièces ne respectent pas les critères de qualité en vigueur, il est possible

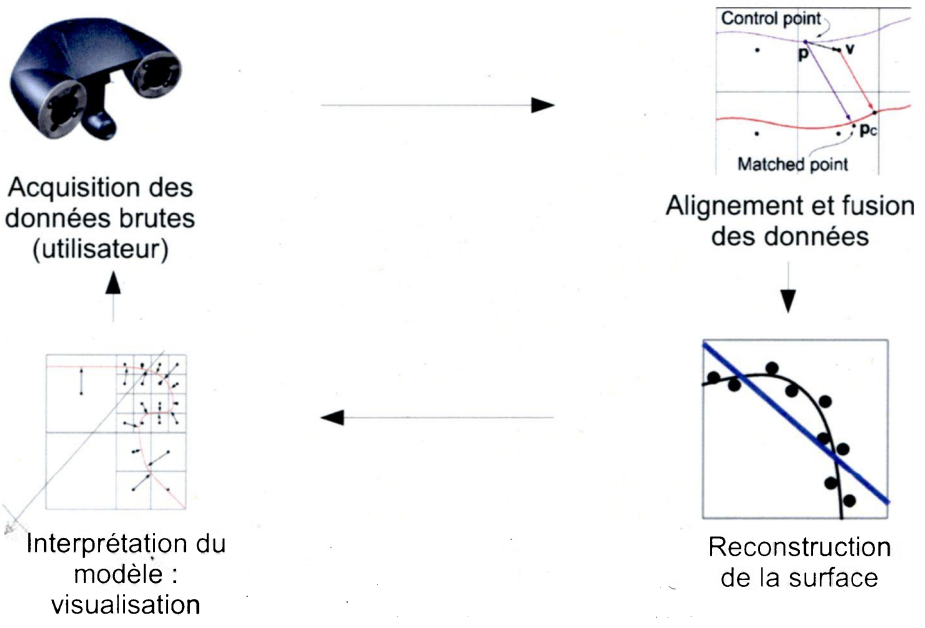


FIG. 1.1 – Boucle de modélisation interactive. La première étape consiste en l'acquisition des données brutes. L'acquisition est contrôlée par l'utilisateur. Viennent ensuite l'alignement des données avec le modèle en cours de reconstruction, la reconstruction de la surface et la visualisation.

de réagir rapidement en arrêtant la production et évitant ainsi des pertes monétaires.

Tout ce qui a trait à la conservation du patrimoine peut également profiter de la modélisation interactive. La modélisation de statues ou de sites archéologiques demande généralement un grand déploiement de matériel. Les utilisateurs sont donc intéressés à valider la qualité des modèles 3D obtenus sur le champ pour déterminer si les modèles sont satisfaisants. Si une erreur s'est glissée lors du processus de reconstruction, il est possible de refaire une acquisition. Avec un système non interactif, le modèle est obtenu en laboratoire et l'utilisateur a peu de recours advenant un problème avec les données brutes [19].

La robotique mobile peut également profiter des avancées faites dans le domaine de la modélisation interactive. Un système automatisé, remplaçant l'utilisateur des précédents exemples, peut avoir à générer en temps réel un modèle 3D de l'environnement dans lequel il se trouve pour lui permettre de planifier ses prochaines actions. Un tel système pourrait être un robot utilisé pour explorer les décombres à la recherche de survivants suite à une catastrophe naturelle. Un autre exemple serait un robot autonome faisant de l'exploration minière dans des endroits dangereux pour les humains.

Les exemples précédents reposent tous sur des systèmes de modélisation interactive

devant solutionner les problèmes suivants en temps réel : l'acquisition des données brutes, l'alignement des données brutes avec le modèle actuel, la fusion des données et la reconstruction de la surface ainsi que l'interprétation du modèle. Ces quatre étapes sont connues sous le nom de boucle de modélisation interactive. Comme le montre le diagramme de la figure 1.1, la boucle de modélisation interactive inclut l'utilisateur. L'utilisateur, humain ou machine, a une influence directe sur le système par le biais de sa stratégie d'acquisition des données brutes en fonction de l'état actuel du modèle. L'état actuel du modèle est fourni par le module d'interprétation des données qui génère à l'utilisateur une version compréhensible de la représentation de l'objet.

La définition de la boucle de modélisation interactive classique ne fait pas intervenir la compression du modèle. La compression du modèle se fait généralement a posteriori [31] : un modèle de l'objet est reconstruit à haute résolution et compressé à la fin de l'acquisition. Le défaut de cette approche est qu'elle nécessite une grande quantité de mémoire, car le modèle doit être reconstruit dans sa totalité à haute résolution avant d'être compressé.

Mes travaux visent donc à pallier à cette lacune en développant un système de modélisation interactive permettant la compression en temps réel de la surface, durant l'acquisition des données brutes. La compression temps réel permettra d'optimiser les ressources disponibles dans le but de faire l'acquisition d'objets de plus grande taille ou à plus haute résolution avec la même quantité de mémoire. Le système proposé repose sur une représentation permettant de reconstruire le modèle de l'objet à différents niveaux de résolution. Le développement de cette représentation multirésolution ainsi que les mécanismes nécessaires à la reconstruction de la surface et sa visualisation sont le coeur de mes travaux de maîtrise. La suite de ce chapitre présente une mise en contexte des avancées scientifiques ayant conduit à la modélisation interactive. Suivra ensuite une explication plus détaillée des objectifs visés par mes travaux.

1.3 Mise en contexte

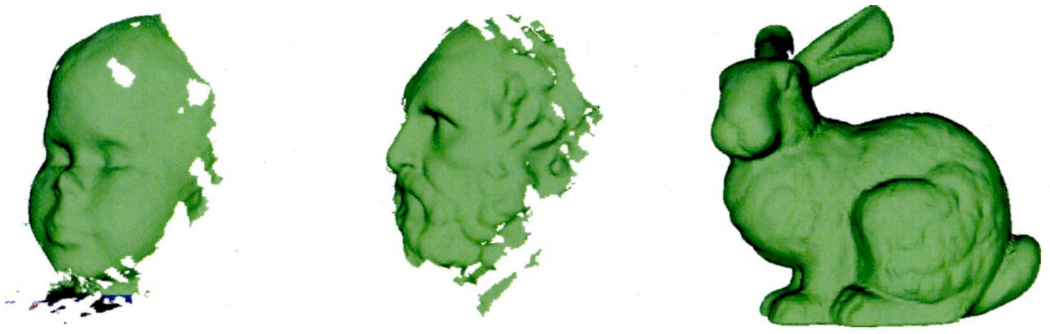
Par le passé, le problème de la modélisation interactive a longtemps été abordé étape par étape. Certains travaux abordèrent uniquement la mise au point d'un capteur [11], d'autres se concentrèrent sur l'alignement des données [5] ou la reconstruction de la surface [14]. Cependant, de récents travaux ont montré que pour solutionner le problème de la reconstruction de surface en temps réel, il n'est pas suffisant de développer des algorithmes performants. Il faut utiliser une représentation de surface adaptée à la tâche. La représentation utilisée est l'élément clé permettant la reconstruction interactive d'un

modèle 3D d'un objet [31].

Parmi les différentes représentations disponibles, telles que les nuages de points, les listes de polygones et les surfaces incluses dans un volume, seules ces dernières semblent adéquates pour résoudre le problème de la modélisation interactive [31]. Cependant, il est important de mentionner que dans de récents travaux [4], une combinaison de différentes représentations a été utilisée avec succès pour obtenir une triangulation d'un nuage de points en temps réel. L'idée consiste à accumuler en chaque point d'une grille régulière dans l'espace une liste de points mesurés dans un voisinage donné. Cette liste de points est ensuite utilisée pour calculer une triangulation locale du modèle. Ce système est intéressant, mais limité pour certaines étapes de la boucle de modélisation interactive, telle que l'alignement des données, et il serait difficile avec cette architecture d'accommoder différentes densités d'acquisition.

Durant la dernière décennie, les représentations volumétriques ont évolué progressivement pour être en mesure de résoudre toutes les étapes de la boucle de modélisation interactive en temps réel. Tout d'abord, Hoppe *et al.* [14] se sont attaqués au problème de la reconstruction d'une surface à partir d'un nuage de points par l'intermédiaire d'une représentation volumétrique. Une fonction implicite de la surface est générée à partir de laquelle un algorithme du type *Marching Cubes* [21] est en mesure d'en extraire une triangulation utilisée à des fins de visualisation. Hilton et Illingworth [13] ainsi que Curless et Levoy [6] ont poussé davantage le développement de la représentation volumétrique dans le domaine de l'imagerie 3D à partir de cartes de profondeur. L'intérêt premier de ces travaux n'est pas la reconstruction temps réel de la surface, mais plutôt la fusion des données brutes et la reconstruction d'une surface de qualité. En améliorant la représentation volumétrique par le biais de l'estimation de la normale à la surface, Rusinkiewicz *et al.* [25] sont en mesure d'aligner des données en provenance d'un capteur 3D et ce, durant l'acquisition des données brutes. Leur système est en mesure d'afficher en temps réel une représentation intermédiaire de la surface. Ce n'est qu'à la fin de l'acquisition, une fois que toutes les données sont recueillies, que le modèle final haute résolution est reconstruit et qu'il est possible d'en obtenir des rendus de qualité.

Tubic *et al.* [32] ont récemment proposé un système basé sur une représentation volumétrique où toutes les étapes de la boucle de modélisation sont intégrées tout en conservant une complexité linéaire en fonction du nombre de points ajoutés au modèle. Le respect de cette contrainte assure que le système restera réactif même après une longue acquisition. En plus d'assurer une complexité linéaire par rapport à la quantité de données brutes, la représentation volumétrique permet l'intégration de différents types de données en provenance de capteurs fournissant des nuages de points, des courbes ou bien des cartes de profondeur sous forme de listes de triangles [31].



(a) (b) **(c)**

FIG. 1.2 – a) et b) Exemples d'objets réels numérisés interactivement avec la représentation vectorielle développée par Tubic [30]. c) Exemple d'un objet reconstruit par le même système à partir de données fournies par l'université Stanford.

Le succès de l'approche proposée par Tubic *et al.* repose sur l'utilisation d'une représentation unique bien adaptée à la modélisation interactive : le champ vectoriel. L'acquisition des données brutes, la mise en registre, la reconstruction de la surface et l'affichage utilisent tous cette même et unique représentation. Simplement, un champ vectoriel est une représentation de surface de type implicite encodant en tout point d'une grille régulière, définie autour de l'objet à modéliser, la distance et la direction du point le plus près sur la surface. Cette particularité solutionne le problème de recherche de points voisins sur la surface encodée qui est à la base de la plupart des opérations nécessaires à la reconstruction de la surface d'un objet. Pour chaque point du volume de travail, le point le plus près sur la surface modélisée est obtenu de façon triviale à partir de l'information inscrite au point de la grille régulière le plus près du point étudié. Aucune recherche exhaustive n'est nécessaire comme dans le cas de l'utilisation d'un nuage de points ou d'une liste de triangles.

Un autre avantage lié à l'utilisation de la représentation vectorielle est la possibilité de solutionner tous les problèmes de la boucle de modélisation interactive avec une représentation unique. L'utilisation d'une représentation unique évite le passage d'une représentation à l'autre pouvant être très coûteux en ressource processeur. Un exemple typique d'un passage d'une représentation à une autre est la génération d'une triangulation à partir d'un champ pour en permettre l'affichage [14].

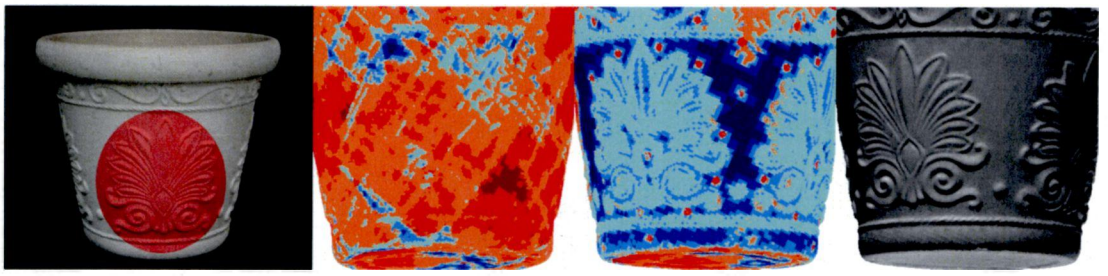
Comme le montre la figure 1.2, le système développé par Tubic [30] a été utilisé avec succès pour faire la reconstruction de plusieurs objets. Avec les capacités d'un ordinateur de table, la taille des objets pouvant être modélisés avec ce système est de l'ordre du mètre cube, avec une résolution de l'ordre du dixième de millimètre.

1.4 Définition du problème et objectifs

Comme le montre les résultats présentés dans [30], un système basé sur une représentation volumétrique est une alternative très intéressante pour la modélisation interactive. L'utilisation d'une représentation unique offrant une complexité linéaire en fonction du nombre de points pour chaque étape de la boucle de modélisation permet d'assurer l'interactivité du système et d'obtenir un modèle de haute qualité, en temps réel, durant l'acquisition des données brutes. Récemment, Hoppe *et al.* ont également opté pour une représentation basée sur les champs vectoriels pour la reconstruction d'objets 3D [16].

Malheureusement, la représentation volumétrique a un défaut de taille : sa grande consommation de mémoire. Jusqu'à récemment, cette lacune empêchait l'utilisation de ce type de système pour l'acquisition d'objets de grande taille et limitait la résolution des modèles. Dans le contexte actuel, un espace de travail de grande taille correspond à un volume de plus d'un mètre cube, toujours avec une résolution de l'ordre du millimètre. Un autre aspect de la représentation volumétrique pouvant être amélioré est la qualité du module permettant l'affichage temps réel de la surface en cours de reconstruction. Les travaux de [30] ont montré qu'il est possible d'implanter un module de visualisation ayant une complexité constante pour une résolution d'image donnée et une taille de volume fixe. Cependant, le rendu fourni par le système proposé n'exploite pas au maximum l'information contenue dans le champ vectoriel et il y a place à améliorations.

L'objectif de mes travaux est donc double. Tout d'abord, développer une représentation vectorielle inspirée de [30] permettant une meilleure gestion des ressources matérielles, plus précisément de la mémoire. Cette gestion avancée de la mémoire se fera par le biais d'une représentation multirésolution permettant la compression du modèle en temps réel, durant l'acquisition des données. Les avantages d'un tel système sont multiples. Tout d'abord, il est possible de modéliser des objets de plus grande taille en conservant la même résolution que dans les travaux précédents ou d'augmenter la résolution du modèle pour un volume de travail similaire à celui utilisé par [30]. De plus, la multirésolution permet d'accélérer l'acquisition d'objets de grande taille en permettant une reconstruction grossière des zones à faible courbure et en concentrant les efforts sur les détails fins. Finalement, il est intéressant de noter que chaque niveau de résolution de la représentation filtre les données brutes différemment, partant d'un filtrage élevé avec les niveaux grossiers, vers un filtrage plus faible associé aux niveaux plus fins. Ces différents niveaux de filtrage permettent de gérer plus efficacement le bruit contenu dans les données brutes.



(a) (b) (c) (d)

FIG. 1.3 – Scénario d'acquisition multirésolution. a) Photo de l'objet. La zone en surbrillance correspond au détail d'intérêt. Modèle représenté en fonction du niveau local de résolution durant b) l'acquisition du contexte et c) l'acquisition des zones d'intérêt. Une teinte foncée indique un niveau de résolution grossier. Une teinte pâle indique un niveau de résolution fin. d) Rendu final du modèle.

Le second objectif de mes travaux est de fournir un rendu de qualité de la surface en cours de reconstruction pour permettre à l'utilisateur de bien évaluer la qualité du modèle durant la séance d'acquisition des données brutes ainsi que la couverture complète de l'objet.

1.4.1 Objectif 1 : Amélioration de la gestion de la mémoire

Pour une même quantité de ressource mémoire, l'acquisition d'objets de plus grande taille passe par une meilleure gestion de la mémoire disponible. L'idée de base pour permettre une telle optimisation des ressources est d'incorporer à la boucle de modélisation interactive un module permettant de choisir *localement* le niveau de résolution optimal pour reconstruire la surface. D'un côté, un niveau de résolution grossier signifie que la reconstruction de la surface se fait dans un volume de travail divisé grossièrement tout en permettant d'encoder de grandes surfaces à faible courbure en utilisant peu de mémoire. Les niveaux grossiers ont également pour fonction de filtrer fortement les données brutes. D'un autre côté, la reconstruction à un niveau de résolution élevé signifie que le volume de travail est divisé finement et qu'il est possible d'encoder une surface contenant beaucoup de détails, mais au prix d'une plus grande consommation de mémoire.

La figure 1.3 montre un exemple d'un scénario d'acquisition multirésolution typique. Ici, l'utilisateur veut modéliser un détail d'intérêt, soit la fleur mise en surbrillance, dans son contexte, soit le reste du vase. La figure 1.3 b) montre l'état du modèle après la première étape de la modélisation du vase, c'est-à-dire une acquisition rapide du

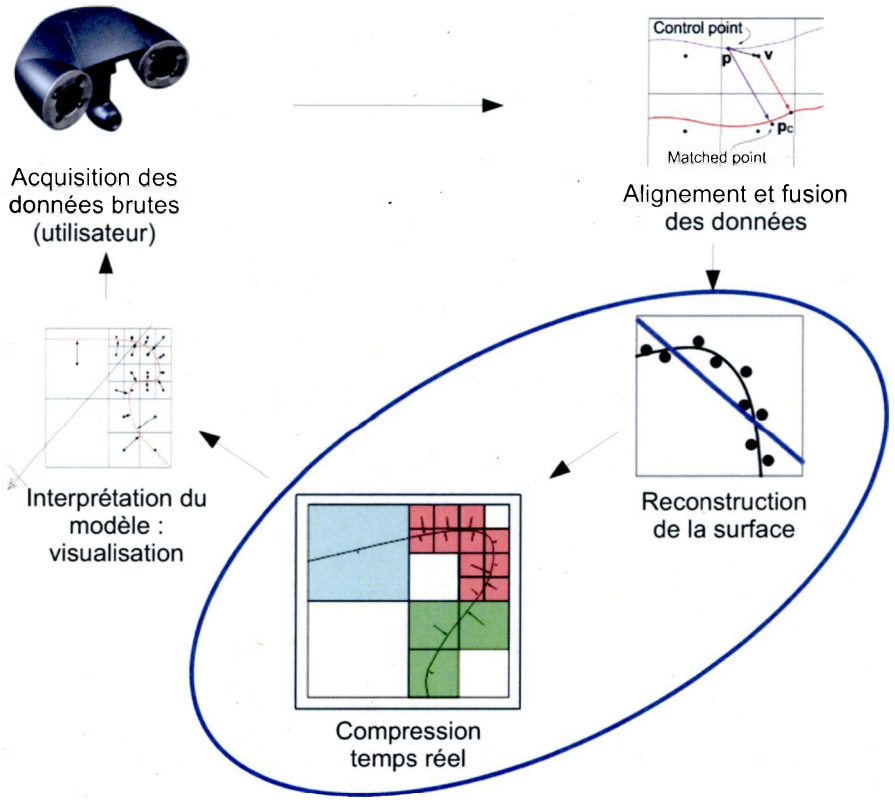


FIG. 1.4 – Structure de la nouvelle boucle de modélisation interactive proposée. L'étape de la compression temps réel a été ajoutée. Le module de sélection du niveau de résolution optimal intervient lors de la *reconstruction de la surface* et de la *compression temps réel* (étapes entourées de bleu). La sélection du niveau de résolution optimal se fait durant l'acquisition des données brutes, de manière interactive.

contexte. Notez qu'une couleur chaude plus foncée correspond à un niveau de résolution grossier. La figure 1.3 c) et d) montre l'état du modèle à la fin de la séance d'acquisition. Notez comme les régions contenant du détail, affichées en bleu clair, ont été reconstruites à un niveau de résolution plus fin tandis que les régions à faible courbure, affichées en bleu foncé, ont été reconstruites à un niveau plus grossier.

La figure 1.4 montre la structure de la nouvelle boucle de modélisation interactive proposée incluant le module de sélection du niveau de résolution optimal de reconstruction de la surface. Ce module affecte l'étape de la *reconstruction de la surface* qui est suivie d'une nouvelle étape, soit *la compression en temps réel*. Dorénavant, dès que l'utilisateur ajoute un certain nombre de données au modèle, la surface est mise à jour en tenant compte des nouvelles règles imposées par le module de sélection du niveau de résolution optimal et, à l'étape de compression en temps réel, le système vérifie s'il

y a des zones pouvant être compressées. Si la surface peut être compressée localement, le système la compresse et libère ainsi de la mémoire qui sera utilisée lors du reste du processus de modélisation, d'où la possibilité de numériser des objets de plus grande taille.

La modification de l'étape de reconstruction et l'ajout de la compression en temps réel dans la boucle de modélisation interactive soulève deux problèmes principaux. Tout d'abord, le système doit permettre la cohabitation de différents niveaux de résolution dans une même représentation volumétrique tout en conservant la cohérence du modèle. Ici, un modèle cohérent est un modèle qui est dépourvu d'artéfacts qui seraient introduits par l'approche de reconstruction. Nous reviendrons sur la notion de *cohérence* de façon plus formelle au chapitre 3. À cause de cette nécessité de faire cohabiter différents niveaux de résolution, la première tâche à accomplir est donc de développer une représentation vectorielle multirésolution.

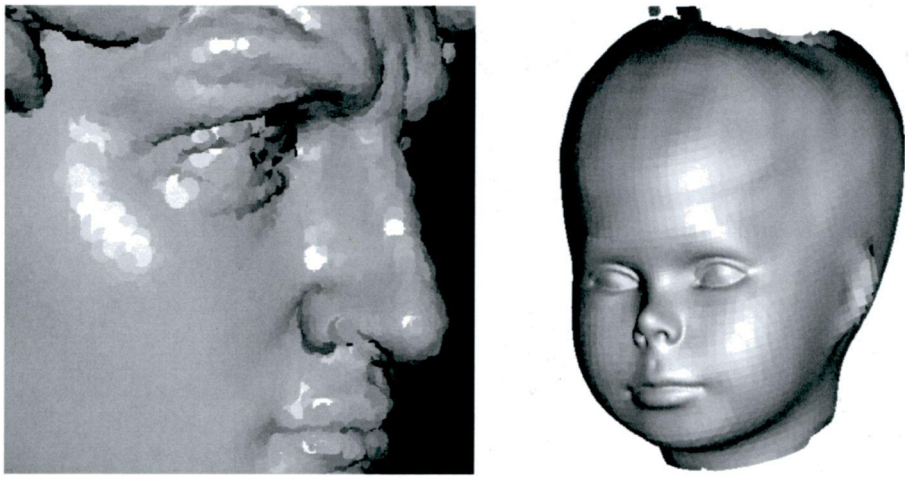
Une fois la représentation vectorielle multirésolution fonctionnelle, la seconde tâche à accomplir est de développer un système permettant de sélectionner *automatiquement* et *localement* le niveau de résolution optimal de reconstruction de la surface pendant l'acquisition des données brutes. Le défi ici est de faire ce choix malgré le flot continu de données en provenance du capteur tout en s'assurant que la mémoire soit utilisée efficacement et que la géométrie de l'objet soit bien captée. Nous verrons que déterminer le niveau de résolution optimal se fait en deux étapes, à partir de la densité de points fournie par l'utilisateur ainsi que du niveau de détail réel de la géométrie de l'objet.

1.4.2 Objectif 2 : Amélioration du module de visualisation

Le but premier du module de visualisation est de permettre de valider la qualité du modèle en temps réel, durant l'acquisition des données. Permettre une telle validation nécessite un module de visualisation de qualité offrant une interprétation fidèle du modèle reconstruit et ce, le plus rapidement possible.

Pour assurer la rapidité des rendus, il n'est pas envisageable de convertir la représentation vectorielle en une autre représentation telle qu'une triangulation. Il faut directement afficher les données contenues dans le champ vectoriel. Pour ce faire, deux techniques de visualisation sont disponibles.

La première famille de techniques permettant d'afficher directement les données contenues dans un champ vectoriel repose sur la projection de pastilles (*splatting*) [26, 34]. Cette technique consiste à parcourir les éléments de données du modèle, cor-



(a) (b)

FIG. 1.5 – Comparaison des rendus obtenus par a) projection de pastilles circulaires (Tirée de [26]) et b) par l'utilisation d'une approche par lancer de rayons (*ray caster*).

respondant aux points de la grille dans le cas d'une représentation volumétrique, et à projeter une pastille pour chaque élément contenant de l'information pertinente sur la surface. La figure 1.5 a) montre un exemple d'un rendu obtenu à partir d'une projection de pastilles.

Une autre famille de méthodes utilisée pour afficher le contenu d'un champ vectoriel repose sur le lancer de rayons (*ray caster*). Le principe consiste à lancer un rayon au travers du volume de travail jusqu'à ce qu'il rencontre la surface. L'orientation de la normale à la surface par rapport au rayon tracé indique l'intensité du pixel affiché. Par le passé, ces techniques ont été abondamment utilisées dans le domaine de l'imagerie médicale [20]. De récents progrès [33] ont permis d'atteindre des vitesses de rendu suffisantes pour envisager un affichage interactif de qualité. La figure 1.5 b) montre un rendu obtenu avec une telle approche.

La question première est donc de déterminer quelle approche est la mieux placée pour remplir le rôle du module de visualisation. Rappelons que le module de visualisation doit permettre de valider la qualité de la reconstruction de la surface en plus d'assurer à l'utilisateur une couverture complète de l'objet. Il sera montré au chapitre 2, suite à l'analyse du pour et du contre de chaque méthode, que l'approche par lancer de rayons est mieux adaptée à la présente tâche.

Une fois cette question réglée, le défi de l'implantation du module de visualisation est d'interpréter de manière cohérente la surface multirésolution encodée dans le champ vectoriel. Les problèmes surviennent généralement à l'interface entre les niveaux de

résolution où il est important de choisir localement le bon niveau de résolution pour assurer un rendu de qualité. Tous les algorithmes nécessaires à la visualisation du modèle seront présentés au chapitre 5.

Malgré le fait que la complexité de l'approche par lancer de rayons soit constante pour un volume donné et une taille d'image fixe, il n'en reste pas moins que la visualisation du modèle est l'étape de la boucle de modélisation interactive qui demande le plus de ressource en terme de puissance de calcul. Pour assurer l'interactivité du système, une approche d'affichage progressive fovéale sera également présentée.

1.5 Plan du mémoire

Ce mémoire explique les étapes menant au développement d'une représentation volumétrique multirésolution permettant d'optimiser les ressources mémoire durant l'acquisition des données brutes tout en offrant à l'utilisateur des rendus de qualité de la surface en cours de reconstruction. Le chapitre 2 est un rappel des notions de base concernant l'utilisation d'un champ vectoriel pour la reconstruction d'une surface 3D de manière interactive. Le chapitre 3 explique le principe derrière l'adaptation de la représentation vectorielle à la multirésolution. Les algorithmes permettant de sélectionner le niveau de résolution optimal en fonction de la densité de points fournie par l'utilisateur et le niveau de détail réel de la géométrie de l'objet sont abordés dans le chapitre 4. Finalement, le chapitre 5 présente le fonctionnement du module interactif de visualisation.

Chapitre 2

Le champ vectoriel pour la reconstruction de surfaces

2.1 Introduction

Avant d'aborder l'objectif principal de mon mémoire qui est de développer une représentation vectorielle profitant d'une gestion avancée de la mémoire par l'utilisation de la multirésolution et de la compression temps réel, il est important de bien comprendre les avantages d'une représentation vectorielle dans le cadre de la modélisation interactive. Les travaux de Tubic [30] ont démontré que pour assurer la réactivité du système de modélisation tout au long de l'acquisition, il ne faut pas seulement développer des algorithmes performants, mais bien utiliser une représentation de surface appropriée. Le champ vectoriel s'avère très performant en ce qui a trait à la modélisation interactive de surface et ce chapitre a pour but d'expliquer les principes de base liés à son utilisation tout en faisant ressortir les avantages de cette représentation.

Tout d'abord, rappelons que pour qu'un système de modélisation 3D soit considéré interactif, il doit conserver sa réactivité peu importe le nombre de données brutes recueillies durant la séance d'acquisition. Cela signifie que l'ajout d'une donnée au modèle existant doit se faire en un nombre constant d'instructions pour une taille de volume donnée, peu importe la géométrie du modèle en reconstruction. D'un point de vue de la complexité algorithmique, l'ajout d'un point au modèle doit donc être de complexité constante ($O(1)$). Il s'avère que le champ vectoriel défini dans un volume de taille fixe permet de respecter cette condition pour l'ajout d'un point au modèle ainsi que pour toutes les autres étapes entrant dans le processus de la modélisation interactive.

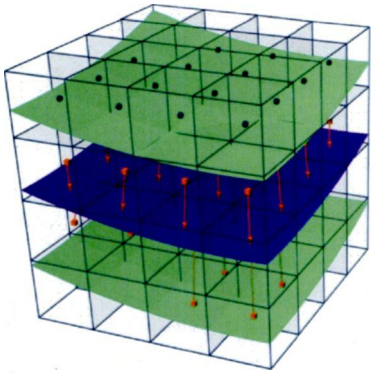


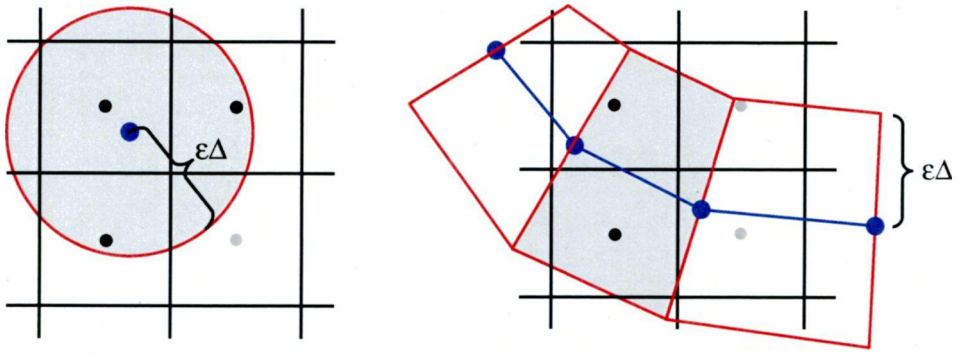
FIG. 2.1 – Exemple d'un champ vectoriel utilisé pour la reconstruction d'une surface. La surface bleue correspond à la surface de l'objet encodée dans le champ. Les deux surfaces vertes correspondent aux limites de l'enveloppe de reconstruction du champ vectoriel. Tirée de [30].

La prochaine section présente une définition formelle de ce qu'est un champ vectoriel dans le contexte de la modélisation interactive. Suivra ensuite un retour sur les étapes de la boucle de modélisation interactive. Il sera expliqué pour chaque étape quel est le rôle du champ vectoriel ainsi que les avantages qu'il apporte. La dernière section du chapitre abordera l'implantation d'un système de reconstruction d'une surface 3D de manière interactive à l'aide d'un champ vectoriel.

2.2 Définition d'un champ vectoriel

Le champ vectoriel fait partie de la famille des représentations implicites de surface. Une représentation implicite, au contraire d'une représentation explicite telle qu'une triangulation, est définie dans un volume englobant l'objet. En chaque point de ce volume, le champ vectoriel encode la distance et la direction du point le plus près sur la surface. La figure 2.1 montre un exemple d'une surface reconstruite à l'aide d'un champ vectoriel. Le volume englobant est séparé en cubes de mêmes dimensions centrés sur les points de la grille régulière. Chacun de ces cubes correspond à un *voxel*. La surface de l'objet est présentée en bleu et passe par les extrémités des vecteurs encodés en chaque point de la grille. Les deux surfaces vertes correspondent à l'enveloppe de reconstruction. Cette dernière permet de limiter la zone de reconstruction du champ vectoriel à une mince couche entourant la surface de l'objet.

De manière plus formelle, un champ vectoriel est une fonction discrète $F(p_v)$ définie sur une grille régulière de points 3D. Pour chaque point p_v de la grille, le champ vectoriel



(a) (b)

FIG. 2.2 – Forme que prennent les cellules fondamentales pour a) un point et b) un segment de courbe.

encode la distance D et la direction $\overrightarrow{N}$ du point le plus près sur la surface de l'objet. Cela correspond à encoder le plan tangent au point le plus près sur la surface. Pour un nuage de points en entrée, le plan tangent à la surface est obtenu à partir d'un moyennage des points 3D bruts tombant dans le voisinage d'un point de la grille. Notez que le champ vectoriel peut également être vu comme une combinaison d'un champ scalaire de distances et de son gradient. Effectivement, en chaque point de la grille, la direction vers la surface (gradient du champ scalaire) ainsi que la distance à cette dernière est encodée. Le détail du calcul de la surface sera présenté à la section 2.4.3.

2.3 Enveloppe de reconstruction et cellule fondamentale

Cette première explication simple de la reconstruction d'un champ vectoriel fait intervenir une notion importante : un voisinage d'un point de la grille. Formellement connu sous le nom d'*enveloppe de reconstruction*, ce voisinage est défini par une sphère de rayon $\varepsilon\Delta$ centrée sur la position du point de la grille en question. La valeur Δ correspond à la taille d'un *voxel* qui est en fait la distance inter-point. Pour sa part, la valeur ε correspond à la taille, en multiple de la taille d'un *voxel*, de l'enveloppe de reconstruction. Toutes les données brutes mesurées dans l'enveloppe de reconstruction d'un point de la grille affectent le plan tangent qui y est encodé.

Lors de l'ajout d'un point au modèle existant, parcourir tous les points de la grille et mettre à jour ceux dont l'enveloppe de reconstruction englobe le nouveau point ajouté serait très lourd et non envisageable dans un contexte de modélisation interactive. Le

processus d'ajout d'un point à la représentation doit donc être vu en sens inverse : lorsqu'un nouveau point est ajouté au volume, les points de la grille qui sont à une distance inférieure à $\varepsilon\Delta$ du nouveau point ajouté sont influencés par ce dernier et doivent être mis à jour. De cette manière, trouver les points de la grille influencés par l'ajout d'une donnée au modèle existant devient trivial. Il est également important de noter que le nombre de points de la grille affectés par l'ajout d'une donnée est borné peu importe la complexité géométrique du modèle reconstruit.

La région définie autour d'une nouvelle donnée est appelée *cellule fondamentale* et est le pendant de l'enveloppe de reconstruction à la différence que la *cellule fondamentale* est centrée sur la donnée et l'enveloppe de reconstruction est centrée sur le point de la grille. Tel que montré à la figure 2.2 a), la cellule fondamentale d'une point 3D correspond à une sphère de rayon $\varepsilon\Delta$ centrée sur la position de ce point. Si le champ est reconstruit avec des données plus structurées, telles que des courbes ou une liste de triangles, les cellules fondamentales prennent respectivement la forme d'un cylindre et d'une pyramide tronquée à base triangulaire [30]. La figure 2.2 b) montre de manière simplifiée la forme que prend la cellule fondamentale d'un segment de droite. Reconstruire le champ vectoriel à partir de structures plus complexes (courbes, triangles) présentent deux avantages importants.

Le premier avantage est que la reconstruction à partir de courbes ou de triangles permet d'exploiter la structure de l'information en entrée indépendamment de la résolution de la grille. Le champ vectoriel est alors une meilleure approximation que le gradient numérique d'un champ scalaire.

Le second avantage de l'utilisation de données plus structurées lors de la reconstruction du champ vectoriel se situe au niveau de la complexité de l'algorithme. Pour bien comprendre cet impact, il faut voir que la complexité algorithmique de la reconstruction du champ vectoriel dépend du nombre de points de la grille devant être mis à jour lors de l'ajout de chaque nouvelle donnée. Dans le cas d'un point 3D, ce nombre correspond aux points de la grille inclus dans la cellule fondamentale associée à ce nouveau point 3D. Celle-ci est une sphère incluant un nombre de points de la grille proportionnel à ε^3. Dans le cas d'une courbe, comme la cellule fondamentale est de forme cylindrique, le nombre de points de la grille devant être mis à jour est proportionnel au volume d'un cylindre qui lui varie en ε^2. Finalement, l'utilisation de triangles génère des cellules fondamentales ayant la forme d'une pyramide tronquée à base triangulaire faisant en sorte que le nombre de points de la grille à mettre à jour est proportionnel à ε.

Malgré les avantages d'utiliser de telles structures complexes, les prochains développements ne seront faits que pour les points pour mettre l'accent sur le principe de recons-

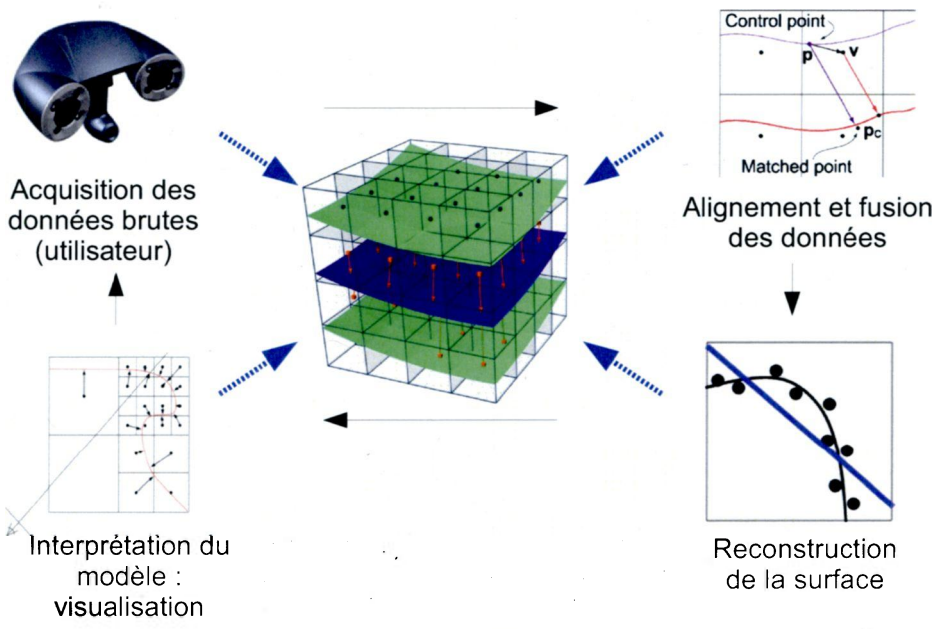


FIG. 2.3 – Boucle de modélisation interactive classique, sans compression. Notez que toutes les étapes de la boucle de modélisation utilisent une représentation unique : le champ vectoriel.

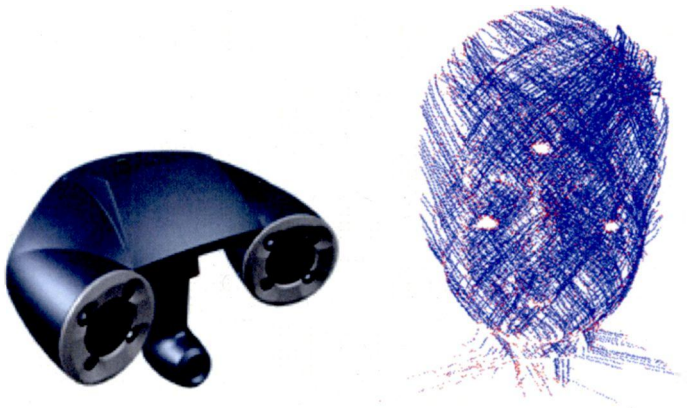
truction et de compression d'un champ vectoriel tout en maintenant les démonstrations à leur état le plus simple et cela, sans perte de généralité. Les principes de reconstruction à partir de courbes ou de triangles sont expliqués en détails dans [30].

2.4 Retour sur la boucle de modélisation interactive

Pour bien comprendre le principe de reconstruction d'une surface d'un objet de manière incrémentale, c'est-à-dire, au fur et à mesure que les données brutes sont ajoutées au modèle, un retour sur la boucle de modélisation interactive sans compression s'impose. Cette section reprendra chacune des étapes de la boucle de modélisation montrées à la figure 2.3 en y expliquant l'utilité du champ vectoriel.

2.4.1 Acquisition des données brutes

Le capteur à la base du système étudié est tenu en main et il utilise un patron laser pour extraire la forme de l'objet [11]. Notez que le système de modélisation in-



(a) **(b)**

FIG. 2.4 – a) Prototype du capteur 3D utilisé pour faire les différentes acquisitions présentées dans ce document. b) Exemple de données brutes obtenues à l'aide de ce type de capteur.

teractive n'est pas spécifique à ce type de capteur. Vous constaterez néanmoins que la modélisation interactive est très bien adaptée à l'utilisation d'un capteur tenu en main.

Configuration du capteur 3D tenu en main

Le capteur est présenté à la figure 2.4 a). La figure 2.4 b) présente un exemple de données brutes obtenues à l'aide de ce type de capteur. La section suivante aborde le fonctionnement de ce capteur en détails.

Ce capteur comprend deux caméras calibrées ainsi qu'une source laser projetant un patron en forme de croix. L'acquisition des images est synchronisée dans le but d'obtenir deux images prises au même moment. Un groupe de deux images est appelé une *trame*. Ce capteur fonctionne selon deux principes fondamentaux pour assurer, dans un premier temps, le positionnement du capteur par rapport à l'objet et, dans un second temps, l'extraction de données 3D.

Positionnenement du capteur 3D tenu en main

Tout d'abord, la paire de caméras stéréo calibrées permet de positionner le capteur à l'aide de marqueurs placés sur l'objet. Ces marqueurs sont utilisés comme points de

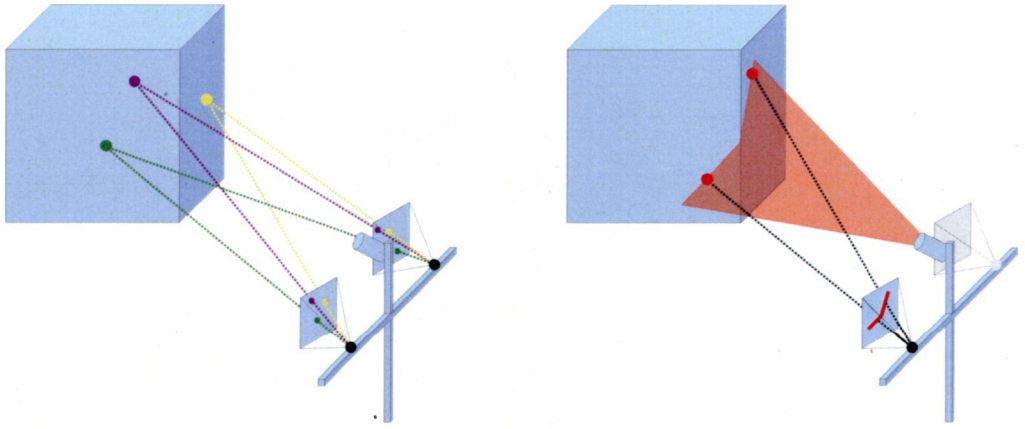


FIG. 2.5 – Principe de fonctionnement du capteur laser tenu en main. a) La position des marqueurs fixes est obtenue par triangulation entre les deux caméras calibrées. b) La trace laser perçue par une caméra correspond à l'intersection du plan laser calibré et de l'objet numérisé.

référence fixes à partir desquels la position du capteur peut être calculée à chaque moment de la séance d'acquisition. Tel que présenté à la figure 2.5 a), lorsqu'un marqueur est visible dans les deux images d'une même trame, sa position 3D est obtenue par triangulation. Étant donné que les caméras sont calibrées, c'est-à-dire que leur position ainsi que leurs paramètres intrinsèques sont connus, cette étape revient à calculer l'intersection entre deux droites dans l'espace. Lorsqu'un modèle 3D des marqueurs est reconstruit pour une trame, il ne reste qu'à le mettre en registre avec le modèle 3D de la position des marqueurs dans un repère global. La transformation obtenue entre les deux nuages de points permet de déterminer la pose du capteur au moment de la prise des images. Il est à noter que le modèle 3D de la position des marqueurs dans un repère global est reconstruit de manière incrémentale durant l'acquisition de l'objet, au fur et à mesure qu'ils sont observés par le capteur.

Pour simplifier le traitement d'image nécessaire à la détection des marqueurs, ces derniers sont faits d'un matériau rétro-réfléchissant et apparaissent avec une forte intensité dans les images lorsqu'ils sont éclairés par une source confondue à l'observateur. Pour ce faire, les caméras sont entourées de diodes électro-luminescentes servant à éclairer les marqueurs. La position obtenue à cette étape permettra d'intégrer les données dans un repère commun.

Reconstruction de points 3D

Une fois la pose du capteur calculée, la reconstruction des points bruts peut débuter. Pour bien comprendre le principe de reconstruction de ce système, il faut voir les traces lasers composant la croix comme des plans lasers dans l'espace. Chacun de ces plans a été préalablement calibré avec une des deux caméras. Tel que montré à la figure 2.5 b), la trace laser perçue par une caméra est en fait l'intersection du plan laser correspondant et de l'objet numérisé.

La première étape de reconstruction des points 3D est d'extraire, pour chaque image, une série de points 2D correspondant à la trace laser. La configuration du capteur fait en sorte que dans chaque caméra, l'orientation du plan laser calibré est verticale. Cela simplifie grandement le traitement d'image, car il suffit d'extraire un point 2D faisant partie de la tracer laser pour chaque ligne de l'image. Par après, comme la caméra est calibrée, il est possible de calculer la position de chacun de ces points 2D dans le plan image. Chacun de ces points définit donc un rayon en provenance du centre de projection de la caméra. La position des points 3D correspond finalement aux points d'intersection de ces rayons et du plan laser. [11].

Les points 3D ainsi obtenus correspondent aux points 3D de l'objet dans le référentiel du capteur. Grâce à la pose du capteur précédemment calculée, il est possible de transformer ces points 3D dans le repère global du modèle et de l'ajouter à la représentation. C'est à ce moment qu'intervient la notion de *cellule fondamentale* pour déterminer quels sont les points de la grille affectés par l'ajout de ces nouvelles données brutes. Tous les points de la grille à l'intérieur de la *cellule fondamentale* de chaque nouvelle donnée doivent être mis à jour.

Le prototype du capteur que nous avons développé au *Laboratoire de Vision et Systèmes Numériques* permet l'acquisition d'environ 1000 points par trame avec une fréquence d'acquisition de 15 trames par seconde. Un tel capteur fournit au système plus de 15000 points par seconde ou 900000 points par minute. L'avantage principal de ce type de capteur est avant tout sa facilité d'utilisation et la possibilité de numériser des objets à géométrie complexe comportant des cavités difficilement accessibles avec un système mécanisé basé sur des étages de translation.

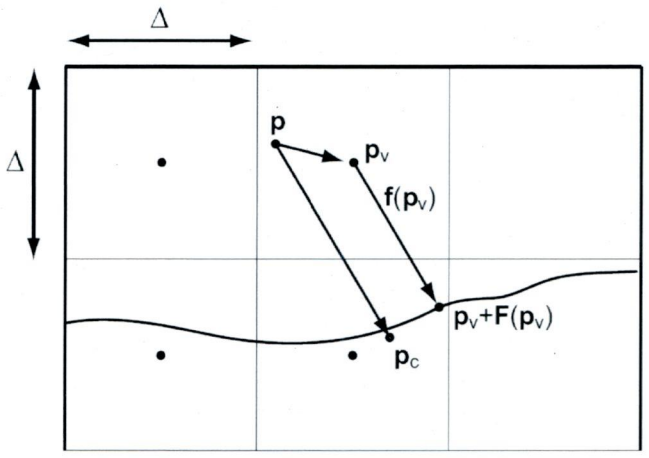


FIG. 2.6 – Approximation du point le plus près p_c d'un point p. Le point p_v correspond au point de la grille le plus près de p. $F(p_v)$ correspond à la valeur du champ vectoriel au point p_v. Δ correspond à la taille des *voxels*, c'est-à-dire, à la distance entre deux points de la grille. Tirée de [30].

2.4.2 Alignement et fusion des données

Lorsque les points 3D des traces lasers sont positionnés dans le repère global du modèle, il faut les aligner et les fusionner avec la surface en cours de reconstruction. L'étape d'alignement et de fusion des données profite grandement de la représentation volumétrique. Pour cette étape de la boucle de modélisation interactive, l'avantage principal du champ vectoriel est la rapidité de la recherche des voisins sur la surface. Effectivement, la recherche du point le plus près sur la surface de n'importe quel point du volume se fait en temps constant, peu importe la complexité géométrique du modèle. Cette particularité avantage grandement les algorithmes d'alignement qui reposent généralement sur une recherche itérative des points les plus près sur la surface par rapport aux données devant être alignées.

Recherche du point le plus près sur la surface

La figure 2.6 montre le principe permettant d'obtenir une approximation du point le plus près sur la surface pour n'importe quel point p situé à l'intérieur du volume de travail. La première étape est de déterminer les coordonnées logiques p_L du point de la grille le plus près du point p. Une coordonnée logique correspond à l'index du point de la grille dans la structure volumétrique. Le point de la grille à l'origine du volume

a une coordonnée logique de $(0, 0, 0)$. Le prochain point de la grille sur l'axe des X a une coordonnée logique de $(1, 0, 0)$ et celui sur l'axe des Z a une coordonnée logique de $(0, 0, 1)$. Pour calculer la coordonnée logique du point de la grille le plus près d'un point p, il suffit de diviser les coordonnées du point p par Δ, soit la taille d'un *voxel* qui correspond également à la distance entre deux points de la grille. Le résultat doit ensuite être tronqué vers le bas, car les coordonnées logiques doivent être entières :

$$p_L = \lfloor p/\Delta \rfloor \tag{2.1}$$

Après avoir obtenu les coordonnées logiques du point de la grille le plus près du point p, il faut trouver les coordonnées physiques de ce point. Cela revient à calculer la position physique, dans le volume de travail, que prend un point logique. Par exemple, le point de la grille ayant les coordonnées logiques $(0, 0, 0)$ est physiquement situé à la position $(\frac{\Delta}{2}, \frac{\Delta}{2}, \frac{\Delta}{2})$. Le prochain point de la grille sur l'axe des X est situé à la position $(\Delta + \frac{\Delta}{2}, \frac{\Delta}{2}, \frac{\Delta}{2})$ et ainsi de suite. Les coordonnées réelles p_v sont obtenues à partir de l'équation suivante :

$$p_v = p_L\Delta + (\frac{\Delta}{2}, \frac{\Delta}{2}, \frac{\Delta}{2}) \tag{2.2}$$

La valeur du champ vectoriel au point p_v, soit $F(p_v)$, est un vecteur pointant vers le point sur la surface qui est le plus près de p_v. La distance du point p_v à la surface correspond donc à la norme de $F(p_v)$. De plus, L'orientation du vecteur $F(p_v)$ correspond à l'orientation de la normale à la surface au point $p_v + F(p_v)$. Ainsi, il est possible d'obtenir la normale à la surface, dénotée $\overrightarrow{N}$, en normalisant $F(p_v)$. Par convention, $\overrightarrow{N}$ est orienté vers l'intérieur de la surface.

$$\overrightarrow{N} = \frac{F(p_v)}{\|F(p_v)\|} \tag{2.3}$$

L'approximation p_c du point le plus près de p est obtenue en projetant sur $\overrightarrow{N}$ le vecteur $\overrightarrow{pp_v}$ et en lui ajoutant $F(p_v)$:

$$p_c = p + \overrightarrow{N} \left\langle \overrightarrow{N}, \overrightarrow{pp_v} \right\rangle + F(p_v) \tag{2.4}$$

Tel que montré à la figure 2.6, le point p_c n'est qu'une approximation de la position

de la surface réelle, car en chaque point p_v de la grille, la surface n'est approximée que par un plan tangent. Donc, si la surface réelle n'est pas réellement plane, une erreur d'approximation est commise. Cette erreur est d'autant plus grande que le point p_c s'éloigne du point $p_v + F(p_v)$.

Mise en registre

Une fois tous les appariements identifiés entre la surface en cours de reconstruction et les points appartenant aux nouvelles traces lasers, l'algorithme de mise en registre de type ICP [1] entre en action. L'idée de cet algorithme est de trouver la transformation rigide qui, une fois appliquée sur les nouvelles traces lasers, minimisera la distance entre les appariements. Si la distance maximale entre les nouveaux points 3D et la surface est au-dessus d'un seuil fixé par l'utilisateur, le processus recommence pour une autre itération.

Entre chaque itération, le champ vectoriel permet de recalculer les appariements très efficacement. Cela signifie qu'après leur transformation, chaque nouveau point 3D est réapparié directement avec son correspondant sur la surface de l'objet et une nouvelle transformation rigide est calculée. Le processus se termine lorsque le déplacement des points 3D de la nouvelle trame tombe sous un certain seuil entre deux itérations consécutives. Lorsque cette condition est respectée, il est alors possible de fusionner les nouvelles données à la surface en cours de reconstruction dans le champ vectoriel en mettant à jour les points de la grille influencés par ces nouvelles données. La procédure exacte de mise à jour est décrite dans la section suivante.

2.4.3 Reconstruction de la surface

Lorsque les données fournies par le capteur sont alignées avec le modèle existant, la reconstruction de la surface proprement dite peut débuter. La *cellule fondamentale* de chaque nouveau point 3D détermine un nombre borné de points de la grille devant être mis à jour. Il est important de noter ici que la détermination des points de la grille situés dans la *cellule fondamentale* d'une donnée revient à faire une recherche des points de la grille situés à une distance inférieure à $\varepsilon\Delta$ de la position de la nouvelle donnée. Les points de la grille respectant cette contrainte peuvent être obtenus en utilisant un principe similaire à l'obtention d'un point le plus près sur la surface, tel que montré à la section 2.4.2. Pour l'ajout de chacun des points p_i au modèle, l'algorithme suivant est utilisé pour déterminer les points de la grille devant être mis à jour :

$p_i \leftarrow$ Nouveau point ajouté à la représentation
$p_L \leftarrow$ coordonnées logiques du point de la grille p_v le plus près de p_i (voir eq. 2.1)
$\Delta \leftarrow$ Taille des *voxels*
$\varepsilon \leftarrow$ Taille de l'enveloppe de reconstruction, en nombre de *voxels*
Pour x **de** $p_{L_x} - \lceil \varepsilon \rceil$ **à** $p_{L_x} + \lceil \varepsilon \rceil$ **faire**
  Pour y **de** $p_{L_y} - \lceil \varepsilon \rceil$ **à** $p_{L_y} + \lceil \varepsilon \rceil$ **faire**
    Pour z **de** $p_{L_z} - \lceil \varepsilon \rceil$ **à** $p_{L_z} + \lceil \varepsilon \rceil$ **faire**
      $p_{physique} \leftarrow \Delta(x, y, z) + (\frac{\Delta}{2}, \frac{\Delta}{2}, \frac{\Delta}{2})$
      $d_{physique} \leftarrow \|p_{physique} - p_i\|$
      Si $(d_{physique} < \varepsilon\Delta)$ **Alors**
        Mettre à jour le point de la grille p_v situé à la position logique (x, y, z)
      Fin Si
    Fin Pour
  Fin Pour
Fin Pour

Algorithme 1: Algorithme utilisé pour mettre à jour le champ vectoriel lors de l'ajout de nouvelles données.

Il est important de noter que la valeur de ε peut ne pas être entière et c'est la raison pour laquelle la recherche se fait dans un voisinage de $\lceil \varepsilon \rceil$ positions logiques autour de la position p_L du point de la grille le plus près de p_i. Malgré le fait que cette approche visite plus de points de la grille qu'il n'en faut, elle a été choisie pour sa simplicité d'implantation. Il suffit de parcourir une triple boucle imbriquée et de ne retenir que les points de la grille dont la distance est inférieure à $\varepsilon\Delta$ du point p_i. Une fois qu'un point de la grille affecté par une nouvelle donnée est connu, sa mise à jour peut débuter.

Reconstruction incrémentale

La modélisation interactive nécessite la reconstruction de la surface de manière incrémentale. Le système doit être en mesure de générer une représentation de la surface de l'objet qui peut être mise à jour avec d'autres données sans toutefois conserver celles obtenues dans le passé. Pour assurer l'interactivité du système et limiter la consommation de mémoire, lorsqu'une donnée est ajoutée au modèle, elle n'est pas conservée. Cela signifie qu'il n'est pas possible, suite à l'ajout d'une donnée, de refaire une estimation globale de la surface sur tous les points ayant tombé dans l'enveloppe de reconstruction d'un point de la grille depuis le début de la séance d'acquisition.

L'approche choisie pour permettre une reconstruction incrémentale est d'encoder, pour chaque point de la grille, une matrice de covariance des points situés dans l'enveloppe de reconstruction ainsi que le centre de masse de ces points. Après avoir extrait les vecteurs et valeurs propres de la matrice de covariance, nous obtenons une estimation de la normale à la surface. Le centre de masse, lui, permet d'évaluer la distance entre le point de la grille et cette estimation de la surface. Le développement suivant, inspiré de [30, 14], montre tout d'abord, pourquoi la matrice de covariance répond aux besoins d'approximation de surface. Par après, il montre que l'ajout d'un point à une matrice de covariance peut bel et bien se faire de façon incrémentale.

Il faut voir l'étape de reconstruction de la surface à partir de points 3D comme étant un processus d'ajustement d'une surface plane sur les points à l'intérieur de l'enveloppe de reconstruction d'un point de la grille. Supposons l'ensemble des points p_i représentant les M points 3D ajoutés au modèle depuis le début de la séance d'acquisition et mesurés dans l'enveloppe de reconstruction d'un point de la grille. Le problème est donc d'estimer le meilleur plan $Ax + By + Cz + D = 0$ passant par ce nuage de points local à partir d'une approche basée sur les moindres carrés.

Pour un point donné p_i, l'erreur commise lors de l'ajustement du plan correspond à la distance d_i de ce point avec la représentation du plan estimé. L'erreur e_i est donc :

$$e_i = d_i = Ap_{i_x} + Bp_{i_y} + Cp_{i_z} + D \tag{2.5}$$

La meilleure approximation d'un plan passant par le nuage de points est donc celle qui minimise la somme des erreurs commises au carré :

$$e = \sum_{i=1}^{M} (Ap_{i_x} + Bp_{i_y} + Cp_{i_z} + D)^2 \tag{2.6}$$

Pour trouver les valeurs des coefficients A, B, C, et D, il suffit de calculer les dérivées partielles de l'équation 2.6 par rapport aux coefficients A, B, C et D et de mettre les équations égales à 0. Nous obtenons les quatre équations normales suivantes :

$$\frac{\partial e}{\partial A} = 2\sum_{i=1}^{M} (Ap_{i_x} + Bp_{i_y} + Cp_{i_z} + D)p_{i_x} = 0, \tag{2.7}$$

$$\frac{\partial e}{\partial B} = 2\sum_{i=1}^{M}(Ap_{i_x} + Bp_{i_y} + Cp_{i_z} + D)p_{i_y} = 0, \tag{2.8}$$

$$\frac{\partial e}{\partial C} = 2\sum_{i=1}^{M}(Ap_{i_x} + Bp_{i_y} + Cp_{i_z} + D)p_{i_z} = 0, \tag{2.9}$$

$$\frac{\partial e}{\partial D} = 2\sum_{i=1}^{M}(Ap_{i_x} + Bp_{i_y} + Cp_{i_z} + D) = 0. \tag{2.10}$$

L'équation 2.10 permet de déduire la valeur du paramètre D qui correspond à la moyenne de la projection des différents points p_i sur la normale du plan. En posant $\overrightarrow{N} = (A, B, C)$ et en s'assurant que $\left\|\overrightarrow{N}\right\| = 1$, D devient :

$$\sum_{i=1}^{M}(Ap_{i_x} + Bp_{i_y} + Cp_{i_z} + D) = 0, \tag{2.11}$$

$$\sum_{i=1}^{M}(Ap_{i_x} + Bp_{i_y} + Cp_{i_z}) = -\sum_{i=1}^{M} D, \tag{2.12}$$

$$\overrightarrow{N}\sum_{i=1}^{M} p_i = -MD, \tag{2.13}$$

$$D = -\overrightarrow{N}\tilde{p}, \tag{2.14}$$

où $\tilde{p}$ correspond au centre de masse de tous les points p_i tombés dans l'enveloppe de reconstruction du point étudié de la grille. Notez que l'équation 2.14 montre que le centre de masse des points fait également partie du plan estimé. Cette observation est très importante. Cela signifie que soustraire le centre de masse à chaque point du nuage revient à centrer ce nuage sur l'origine du système de coordonnées et facilite grandement la tâche de solutionner les paramètres A, B, et C. Les équations 2.7, 2.8 et 2.9 peuvent maintenant être réécrites sous la forme suivante :

$$\underbrace{\frac{1}{M}\sum_{i=1}^{M}\begin{bmatrix} (p_{x_i} - \tilde{p}_x)(p_{x_i} - \tilde{p}_x) & (p_{y_i} - \tilde{p}_y)(p_{x_i} - \tilde{p}_x) & (p_{z_i} - \tilde{p}_z)(p_{x_i} - \tilde{p}_x) \\ (p_{x_i} - \tilde{p}_x)(p_{y_i} - \tilde{p}_y) & (p_{y_i} - \tilde{p}_y)(p_{y_i} - \tilde{p}_y) & (p_{z_i} - \tilde{p}_z)(p_{y_i} - \tilde{p}_y) \\ (p_{x_i} - \tilde{p}_x)(p_{z_i} - \tilde{p}_z) & (p_{y_i} - \tilde{p}_y)(p_{z_i} - \tilde{p}_z) & (p_{z_i} - \tilde{p}_z)(p_{z_i} - \tilde{p}_z) \end{bmatrix}}_{C(v)} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.15}$$

À partir de cette étape, la matrice de covariance devient clairement visible. Notez que la division de la matrice par $\frac{1}{M}$ n'influence pas le résultat, car la partie de droite

est nulle. Sous format plus compact, la matrice de covariance prend donc la forme de :

$$C(v) = \frac{1}{M}\sum_{i=1}^{M}(p_i - \widetilde{p})(p_i - \widetilde{p})^T. \tag{2.16}$$

C'est en développant 2.16 que la forme incrémentale du problème devient visible. Cette nouvelle forme montre qu'il est possible de reconstruire incrémentalement $C(v)$ en ne sauvegardant qu'un petit nombre de données :

$$C(v) = \frac{1}{M}\sum_{i=1}^{M}(p_i - \widetilde{p})(p_i - \widetilde{p})^T = \frac{1}{M}\sum_{i=1}^{M}(p_i p_i^T - \widetilde{p}\widetilde{p}^T) \tag{2.17}$$

avec

$$\widetilde{p} = \frac{1}{M}\sum_{i=1}^{M} p_i. \tag{2.18}$$

Avec cette approche, il n'est pas nécessaire d'emmagasiner toutes les anciennes données pour mettre à jour la matrice de covariance. Il suffit maintenant de conserver les sommes de produits suivantes :

$$\begin{aligned} S_{xx} &= \sum_{i=1}^{M} p_{i_x} p_{i_x} \quad S_{yy} = \textstyle\sum_{i=1}^{M} p_{i_y} p_{i_y} \quad S_{zz} = \sum_{i=1}^{M} p_{i_z} p_{i_z} \\ S_{xy} &= \sum_{i=1}^{M} p_{i_x} p_{i_y} \quad S_{xz} = \textstyle\sum_{i=1}^{M} p_{i_x} p_{i_z} \quad S_{yz} = \sum_{i=1}^{M} p_{i_y} p_{i_z} \end{aligned} \tag{2.19}$$

Il faut également sauvegarder le centre de masse des points. Il suffit de sommer les vecteurs $\overrightarrow{pp_v}$ de cette manière :

$$(S_x, S_y, S_z) = \sum_{i=1}^{M} \overrightarrow{pp_v} \tag{2.20}$$

Lorsque le centre de masse est nécessaire, il suffit de diviser (S_x, S_y, S_z) par le nombre

de points M. À partir des six éléments de l'équation 2.19 et des trois éléments de l'équation 2.20, il est possible de calculer la normale à la surface en reconstruisant la matrice de covariance de la manière suivante :

$$C(v) = \begin{bmatrix} \frac{1}{M}S_{xx} - (\frac{1}{M}S_x)^2 & \frac{1}{M}S_{xy} - \frac{1}{M}S_x\frac{1}{M}S_y & \frac{1}{M}S_{xz} - \frac{1}{M}S_x\frac{1}{M}S_z \\ \frac{1}{M}S_{xy} - \frac{1}{M}S_x\frac{1}{M}S_y & \frac{1}{M}S_{yy} - (\frac{1}{M}S_y)^2 & \frac{1}{M}S_{yz} - \frac{1}{M}S_y\frac{1}{M}S_z \\ \frac{1}{M}S_{xz} - \frac{1}{M}S_x\frac{1}{M}S_z & \frac{1}{M}S_{yz} - \frac{1}{M}S_y\frac{1}{M}S_z & \frac{1}{M}S_{zz} - (\frac{1}{M}S_z)^2 \end{bmatrix} \tag{2.21}$$

Une fois reconstruite, il suffit d'obtenir les vecteurs et les valeurs propres de la matrice de covariance. Il est important de noter que par sa structure, la matrice de covariance est définie symétrique et strictement positive. Cela facilite grandement l'extraction des valeurs propres qui sont toujours réelles. Le vecteur propre associé à la plus petite valeur propre correspond au vecteur normal ($\vec{N}$) au plan tangent encodé. Ce plan tangent représente le nuage de points mesuré dans l'enveloppe de reconstruction du point étudié de la grille et sa normale correspond également à la normale à la surface la plus près de ce point de la grille. La distance est obtenue en projetant le centre de masse sur ce vecteur normal :

$$D = \left\langle \vec{N}, \frac{1}{M}(S_x, S_y, S_z) \right\rangle \tag{2.22}$$

Taille de l'enveloppe de reconstruction

Un des principaux paramètres sur lequel l'utilisateur peut jouer pour influencer le comportement du processus de reconstruction est la taille de l'enveloppe de reconstruction. Cette dernière doit être fixée en fonction de différents facteurs tels que le bruit et la densité des données brutes.

Un des facteurs les plus importants devant être pris en compte lors du choix de la taille de l'enveloppe de reconstruction est l'amplitude du bruit contenu dans les données du capteur. L'utilisation d'une grande enveloppe de reconstruction fait office de filtre passe-bas sur les données, car ces dernières sont moyennées sur un plus grand voisinage. Cela a pour effet de diminuer localement l'impact du bruit sur la reconstruction de la surface. Cependant, il ne faut pas oublier que si une grande enveloppe de reconstruction réduit l'influence du bruit, elle réduit également la possibilité de reconstruire des objets ayant des arêtes vives, car elle filtre les détails. Le choix de la taille de l'enveloppe est

donc un compromis entre le niveau de filtrage du bruit et la résolution du plus petit détail pouvant être reconstruit. En pratique, pour des données très bruitées, la taille de l'enveloppe de reconstruction doit être fixée aux environs de 2Δ ou 3Δ. Si le capteur fournit des points de meilleure qualité, la taille de l'enveloppe peut descendre aussi bas que 1Δ et même $\frac{\sqrt{3}}{2}\Delta$, qui correspond à la demi-diagonale d'un *voxel* et qui est en fait la taille minimale de l'enveloppe pour assurer le recouvrement complet de la surface reconstruite par l'union des enveloppes de reconstruction.

Validité de la surface

Un autre facteur influençant la taille de l'enveloppe est la densité d'acquisition des données. Pour une faible densité de points, l'enveloppe de reconstruction doit être fixée à une taille élevée pour qu'un nombre suffisant de points participent à l'estimation du plan tangent encodé. Le nombre minimum de points requis pour considérer les données de la matrice de covariance valide est fixé de manière empirique, mais doit être au minimum 3. Pour des données très bruitées, il peut arriver que même si un grand nombre de points participent au calcul de la matrice de covariance, la représentation du champ soit invalide, c'est-à-dire que le plan tangent présente une orientation quelconque à cause du bruit. Cela nous amène donc à définir un point de la grille d'un *champ vectoriel valide* de la façon suivante :

Champ vectoriel valide : La valeur $F(p_v)$ du champ vectoriel en un point p_v de la grille est *valide* si et seulement si les données mesurées dans l'enveloppe de reconstruction depuis le début de l'acquisition décrivent un plan tangent avec une erreur inférieure à un certain seuil correspondant à une fraction de son étendue.

Cette définition signifie que la valeur du champ vectoriel en un point de la grille doit représenter un plan tangent, c'est-à-dire que les points mesurés dans l'enveloppe doivent être distribués de façon à former un plan ayant une certaine étendue et une épaisseur faible comparée à cette étendue. L'épaisseur représente en fait le niveau de bruit des données brutes utilisées pour reconstruire le champ vectoriel à cet endroit.

Pour déterminer si les données situées dans l'enveloppe de reconstruction ont généré un plan tangent valide, un test est effectué sur les valeurs propres (e_1, e_2, e_3) de la matrice de covariance. Ce test consiste à normaliser $(e_1 + e_2 + e_3 = 1)$, à classer en ordre croissant les valeurs propres obtenues $(e_1 < e_2 < e_3)$ et à vérifier que les deux

(a) (b)

FIG. 2.7 – Influence de la taille de l'enveloppe de reconstruction pour des données bruitées. a) Taille d'enveloppe de $\varepsilon = 1$ et b) de $\varepsilon = 2$.

plus grandes sont de valeurs non négligeables et que la plus petite est considérablement plus petite que les deux autres :

$$e_2 > 0.05, \tag{2.23}$$

$$e_1 < 0.5e_2. \tag{2.24}$$

Comme le montre l'inéquation 2.23, un plan tangent valide nécessite deux valeurs propres suffisamment importantes (e_2 et e_3) pour assurer une distribution des points bruts mesurés principalement selon deux axes représentant le support d'un plan. De plus, l'inéquation 2.24 montre que l'épaisseur de la distribution des points bruts doit être faible par rapport à l'étendue du plan tangent pour assurer un niveau de bruit acceptable [30]. Si ce critère n'est pas respecté, la valeur du champ vectoriel à ce point de la grille ne peut être considérée valide, car la distribution des points prend plutôt la forme d'un ellipsoïde qu'un plan et l'orientation de la surface est inutilisable.

La figure 2.7 montre un exemple de l'acquisition d'un objet synthétique pour deux tailles d'enveloppe de reconstruction. Ce jeu de données a été obtenu à partir du simulateur d'un capteur 3D projetant une seule trace laser verticale sur l'objet. Le jeu de données comprend un total de 3 millions de points et la grille a une résolution maximale de 512 points par côté. Un bruit uniforme de 0.8Δ a été ajouté aux points de 5% des traces pour simuler une erreur de positionnement du capteur. La figure 2.7 a) montre le résultat de la reconstruction pour une enveloppe de taille $\varepsilon = 1$. Dans ces conditions, la surface montre des discontinuités et, à certains endroits, la surface n'a pas pu être

reconstruite adéquatement. Cela est dû en grande partie à la faible quantité de points mesurés dans chaque enveloppe de reconstruction. Certains trous sont également dus au fait que les matrices de covariance ont échoué le test de validité. Les éléments de surface semblant flotter au dessus de l'oeil gauche de la tête du mannequin sont dus au fait qu'un point de la grille loin de la surface a été en mesure de capter suffisamment de points bruts pour passer le test de validité. Cela a comme résultat de générer des éléments de surface flottants au-dessus de la surface réelle de l'objet.

Tel que montré à la figure 2.7 b), le problème est complètement réglé pour une taille d'enveloppe de $\varepsilon = 2$. Augmenter davantage la taille de l'enveloppe n'apporte pas de gain au niveau de la reconstruction de la surface et est à éviter, car cela a pour effet de filtrer davantage les détails de l'objet.

Cet exemple montre bien le compromis devant être fait entre la taille de l'enveloppe et le niveau de bruit des données. Une taille d'enveloppe inutilement grande diminue le niveau de détail pouvant être reconstruit fidèlement et une taille d'enveloppe trop petite en fonction du niveau de bruit entraînera des discontinuités déplaisantes.

2.4.4 Visualisation

La dernière étape de la boucle de modélisation est la visualisation de la surface pour permettre à l'utilisateur de visualiser l'évolution de la reconstruction en temps réel. L'utilité du module de visualisation est double. Tout d'abord, il permet à l'utilisateur de s'assurer qu'il a complètement couvert l'objet avec ses données et que le modèle final ne contiendra pas de trous. La seconde utilité de ce module est de permettre à l'utilisateur de valider la qualité de la reconstruction en cours. Valider la qualité de la reconstruction durant l'acquisition des données permet de détecter rapidement tout problème. Cependant, la validation de la qualité de la reconstruction nécessite un module de visualisation offrant un rendu fidèle de la surface encodée dans le volume.

Le module de visualisation profite également de la représentation vectorielle de surface pour conserver son interactivité tout au long de la séance d'acquisition. Durant l'acquisition, le nombre de points 3D bruts augmente continuellement. Cela signifie que si le module de visualisation était obligé d'afficher chacun de ces points individuellement, le système écraserait tôt ou tard. Comme la représentation volumétrique combine les points mesurés dans chaque enveloppe de reconstruction peu importe leur nombre, le nombre d'éléments nécessaires pour représenter la surface tend à se stabiliser au cours de la séance d'acquisition et facilite grandement la tâche du module de visualisation.

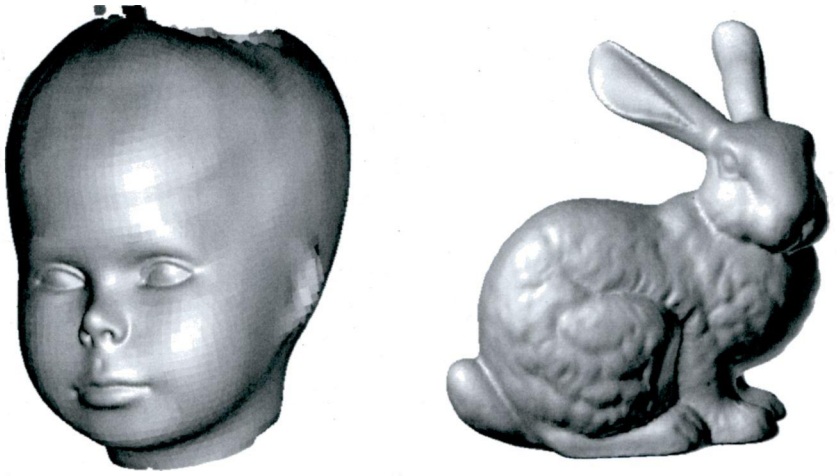
(a) (b)

FIG. 2.8 – Différentes techniques de projection de pastilles. a) Projection de pastilles circulaires uniquement. Tirée de [26] b) Projection de noyaux gaussiens elliptiques dont les orientations sont déterminées en fonction des normales à la surface par rapport au point de vue de l'observateur. Tirée de [34].

Pour conserver l'avantage de la fusion des données dans le champ vectoriel, le module de visualisation doit utiliser une technique de visualisation utilisant directement les données contenues dans la représentation volumétrique. Convertir les données contenues dans le champ dans une autre représentation, telle qu'une triangulation, n'est pas envisageable.

Projection de pastilles

L'affichage des données du champ vectoriel peut se faire selon deux techniques différentes. La première famille de techniques permettant d'afficher les données contenues dans un champ vectoriel repose sur la *projection de pastilles* (*splatting*) [26, 34]. Ces techniques consistent à parcourir tous les éléments de données du modèle, correspondant aux points de la grille dans le cas présent, et à projeter une pastille pour chaque point de la grille contenant de l'information valide sur la surface. La forme de la pastille projetée est généralement un cercle de taille suffisante pour éviter l'apparition de trous entre les pastilles adjacentes. Tel que montré à la figure 2.8 a), cette façon de faire confère une apparence d'écailles de poisson au modèle final [26], empêchant l'utilisateur de bien apprécier la qualité de la reconstruction du modèle. Une autre technique montrée à la figure 2.8 b) utilise un noyau gaussien de forme elliptique dont l'orientation dépend de la normale à la surface [34]. Cette approche offre un rendu de plus grande qualité, mais requiert une gestion avancée des données pour éviter la



(a) (b)

FIG. 2.9 – Rendus obtenus avec des techniques d'affichage basées sur le lancer de rayons. a) Modèle d'une tête de mannequin. b) Modèle reconstruit à partir des données du *Bunny* de l'université Stanford.

présence de trous entre les pastilles, la rendant difficilement utilisable dans un contexte interactif. L'avantage de la projection des pastilles repose sur la rapidité d'affichage lorsque le nombre d'éléments à afficher est peu élevé et qu'il est difficile d'organiser les données dans l'espace. Ces techniques ont fortement été utilisées pour l'affichage de nuages de points bruts, car avec une telle représentation, il n'est pas possible d'identifier facilement les points voisins dans le but de structurer les données. Malheureusement, lorsque le nombre d'éléments à afficher devient trop élevé, l'algorithme écrase et n'est plus utilisable. Il ne faut pas oublier que la complexité de la projection de pastilles est linéaire avec le nombre d'éléments à afficher ($O(n)$). La solution utilisée dans [26] pour contourner ce problème est d'organiser hiérarchiquement le nuage de points pour qu'un nombre constant d'éléments soient affichés à chaque instant. Lorsque l'objet est visible en entier, seulement les niveaux grossiers des pastilles sont affichés. Lorsque l'utilisateur applique un zoom sur une partie de l'objet, un niveau plus fin de pastilles est utilisé, mais sur une zone plus petite de l'objet. Le défaut de cette approche est la nécessité d'un post-traitement pour générer la structure hiérarchique qui peut prendre plusieurs heures à s'exécuter. Cette possibilité n'est pas envisageable dans un contexte de modélisation interactive.

Le lancer de rayons

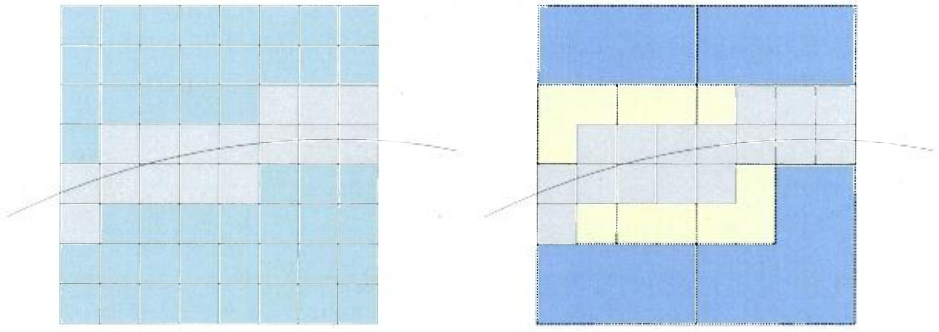
Une autre méthode utilisée pour afficher le contenu d'un champ vectoriel est l'utilisation de la technique de *lancer de rayons* (*ray caster*). Le principe consiste à lancer

un rayon au travers du volume de travail jusqu'à ce qu'il rencontre la surface et ce, pour chaque pixel du plan image. Une surface est détectée lorsque le rayon, en progressant dans le volume, rencontre un passage par zéro de la norme du champ vectoriel. L'orientation de la normale à la surface par rapport au rayon tracé indique l'intensité du pixel affiché. La figure 2.9 montre la qualité des rendus pouvant être obtenue avec une telle approche. Notez que le phénomène de facettes de la figure 2.9 a) n'est pas dû à une limitation de l'approche par lancer de rayons, mais plutôt à la reconstruction de la surface du mannequin à un niveau de résolution grossier. La figure 2.9 b) montre un modèle reconstruit à partir des données du *Bunny* de l'université *Stanford*.

Suite à l'analyse du pour et du contre des deux méthodes présentées pour l'affichage interactif de la surface, la méthode retenue est basée sur le lancer de rayons et ce, pour deux raisons. La raison principale réside en la grande qualité de rendu pouvant être atteinte avec cette méthode comparée aux techniques basées sur la projection de pastilles. Un but important du module de visualisation interactif est de permettre à l'utilisateur de valider la qualité du modèle durant la reconstruction de la surface et les techniques basées sur la projection de pastilles n'offrent pas la qualité recherchée dans un contexte de modélisation interactive. Pour atteindre la qualité recherchée avec une approche par projection de pastilles, il faudrait utiliser une technique avancée telle que décrite en [34] qui nécessite absolument un traitement post-acquisition qui n'est pas permis dans le contexte actuel.

La seconde raison penchant en la faveur de l'approche par lancer de rayons est sa complexité algorithmique constante pour une taille de volume et une dimension d'image données. Cela signifie que peu importe la complexité de la surface reconstruite, le temps d'affichage d'une image est borné pour des résolutions d'image et de volume fixées. Cela assure que le système n'écrasera pas durant la reconstruction de la surface. Il est certain qu'au début de l'acquisition, l'approche par projection de pastilles est plus rapide que celle basée sur le lancer de rayons, car le nombre d'éléments à afficher est faible. Cependant, lorsque la géométrie de la scène se complexifie, le lancer de rayons devient plus avantageux que la projection de pastilles à cause de sa complexité constante.

Comme le module de visualisation basé sur le lancer de rayons est une étape complexe de la boucle de modélisation, le chapitre 5 est réservé exclusivement à l'explication de son fonctionnement.



(a) (b)

FIG. 2.10 – Exemple 2D montrant la différence entre a) une grille complète et b) un *octree* pour la modélisation d'une section de surface. Dans le cas de la grille complète, 16 points de la grille sont utilisés pour encoder la surface (en gris) et 48 points de la grille sont utilisés pour consolider la structure de données. Dans le cas de l'*octree*, 16 noeuds feuilles sont utilisés pour encoder la surface et seulement 12 noeuds sont utilisés pour consolider la structure de données.

2.5 Implantation monorésolution

Maintenant que le côté théorique de la reconstruction d'une surface à l'aide d'un champ vectoriel a été abordé, présentons l'approche qui est utilisée pour son implantation. L'observation majeure qui a guidé le choix de la structure de données utilisée pour l'implantation du champ vectoriel est que le système fait l'acquisition de la surface d'un objet, une structure pouvant être considérée à deux dimensions, dans une représentation volumétrique, qui elle, est à trois dimensions. Cela signifie que la majeure partie du volume est vide. Dans ce contexte, le nombre de points de la grille contenant une surface valide est proportionnel à l'aire de la surface. Cette observation montre qu'implanter un champ vectoriel en allouant en mémoire la totalité de la grille serait un gaspillage non négligeable de ressources et limiterait le système à l'acquisition d'objets de petite taille. Il faut donc une structure de données qui évite l'utilisation de mémoire lorsque certaines zones du volume sont vides.

La structure de données idéale respectant cette ligne de pensée est un arbre binaire étendu en trois dimensions. Généralement connue sous le nom d'*octree*, cette structure de données est très efficace au niveau mémoire, car elle évite l'allocation de ressources pour les régions ne contenant pas d'information. Cette structure de données a grandement été utilisée pour résoudre différents problèmes passant de la génération de triangulations [3] jusqu'à l'appariement de structure à l'aide de transformations non rigides [28].

Variable	Type	Taille (octets)
Distance	réel	4
Centre de masse	vecteur3D	12
Nombre de points	entier	4
Covariance	réel[6]	24
Normale	vecteur3D	12
Position du capteur	vecteur3D	12
Total		68

TAB. 2.1 – Structure de données utilisée pour sauvegarder la représentation de la surface en chaque point de la grille situé à proximité de la surface de l'objet.

La figure 2.10 montre l'avantage frappant de l'*octree* comparé à une grille régulière. Le principe d'un *octree* est que pour chaque incrément de niveau de résolution, le volume est divisé en deux selon chaque axe, résultant en la création de huit noeuds enfants. Cette subdivision est faite de manière récursive jusqu'à ce que le niveau de travail soit atteint. Ce niveau est fixé en fonction du nombre de subdivisions voulues pour l'acquisition en cours. Un niveau de 7 divisera le volume en $2^7 = 128$ parties égales sur chaque axe. Un niveau de 8 donnera $2^8 = 256$ subdivisions et ainsi de suite. Notez que seul le dernier niveau de l'*octree* contient de l'information au sujet de la surface. Les niveaux intermédiaires ne servent qu'à faire le lien entre le noeud racine et les noeuds feuilles pour ainsi éviter d'allouer de la mémoire aux endroits du volume de travail ne contenant pas d'information sur la surface. Dans l'exemple de la figure 2.10, l'*octree* utilise 16 noeuds feuilles pour encoder la surface (en gris) et 12 noeuds pour consolider la structure de données. Dans le cas de la grille régulière, il y a toujours 16 points de la grille utilisés pour encoder la surface, mais 48 points de la grille inutilisés consomment également de la mémoire pour consolider la structure de données.

Dans la représentation vectorielle, chaque point de la grille situé à proximité de la surface de l'objet encode un plan tangent en décrivant une section. Concrètement, cela signifie que chacun de ces points de la grille doit contenir une structure de données emmagasinant les informations nécessaires à la sauvegarde de la représentation de la surface. La table 2.1 montre la structure de données utilisée à chaque point de la grille valide pour encoder l'information concernant la surface. Le *centre de masse* correspond à l'accumulation des vecteurs partant du point de la grille et pointant vers les données 3D situées dans l'enveloppe de reconstruction du point de la grille. Lorsque la *distance* est nécessaire, il suffit de diviser ce vecteur par le *nombre de points* ayant été mesurés dans l'enveloppe de reconstruction et projeter le tout sur la *normale*. Par souci d'économie de puissance de calcul, la *distance* est emmagasinée même s'il était possible de toujours la calculer à partir de l'information contenue dans le *centre de masse*, la *normale* et le *nombre de points*. La variable *covariance* encode les 6 éléments nécessaires

Modèle	Grille régulière	*Octree*
Tête de mannequin	$512^3 * 4 + 1499805 * 68 = 638,9$Mo	$522951 * 32 + 1499805 * 68 = 118,7$Mo
Bunny de Stanford	$512^3 * 4 + 1579334 * 68 = 644,3$Mo	$481420 * 32 + 1579334 * 68 = 122,8$Mo

TAB. 2.2 – Consommation de mémoire pour deux modèles différents reconstruits à l'aide d'un champ vectoriel encodé dans une grille régulière complète et dans un *octree*.

à la reconstruction de la matrice de covariance utilisée pour calculer la *normale*. Finalement, il faut stocker la moyenne des *positions du capteur* au moment de l'acquisition des données pour être en mesure d'identifier le côté intérieur et le côté extérieur de la surface, information nécessaire pour afficher correctement le modèle. Au total, chaque point de la grille à proximité de la surface nécessite 68 octets.

Pour calculer la consommation de mémoire totale pour les deux approches étudiées, il faut tenir compte de la mémoire utilisée pour encoder la surface et la mémoire utilisée pour consolider la structure de données. Dans le cas d'une grille complète, l'approche la plus économique est d'utiliser un pointeur pour chaque point de la grille et de n'allouer la mémoire qu'aux points de la grille situés dans l'enveloppe de la surface. Cela signifie que les points de la grille autour de la surface de l'objet utiliseront $68+4 = 72$ octets et les points de la grille situés dans le vide n'utiliseront que 4 octets. La première colonne de la table 2.2 montre la consommation totale de mémoire pour cette technique pour deux modèles différents reconstruits dans une grille de 512^3 points. L'équation utilisée est la suivante :

$$Memoire = PointsGrille_{Total} * 4 + PointsGrille_{Surface} * 68 \quad (2.25)$$

Dans le cas de l'*octree*, chaque noeud feuille occupe 68 octets de mémoire. Les noeuds intermédiaires servant à consolider la structure de l'*octree* occupent $8 * 4 = 32$ octets correspondant aux 8 pointeurs utilisés pour faire le lien entre un noeud parent et ses noeuds enfants. La consommation de mémoire pour les deux mêmes modèles encodés avec un *octree* de 9 niveaux de profondeur, soit une résolution de 512 éléments par côté, est présentée dans la deuxième colonne de la table 2.2. L'équation utilisée pour calculer la consommation mémoire totale pour un *octree* est :

$$Memoire = Noeuds_{Intermediaires} * 32 + Noeuds_{surface} * 68 \quad (2.26)$$

Cette simple comparaison montre que l'utilisation d'un *octree* est essentielle au bon fonctionnement du système de modélisation interactive sur des ordinateurs de table conventionnels. Avec une résolution de 512 points par côté, l'*octree* permet une économie mémoire de plus de 80% comparé à la grille régulière. Ce gain de mémoire risque même d'être beaucoup plus élevé au fur et à mesure que la dimension du volume augmente.

2.6 Conclusion

Ce chapitre a fait un survol du fonctionnement d'un système de modélisation interactive utilisant une représentation de surface basée sur le champ vectoriel. Il a été montré que chaque étape de la boucle de modélisation interactive profite grandement des avantages liés à cette représentation, car chacune d'entre elle peut être exécutée en temps constant. Il a également été montré qu'au niveau de la consommation mémoire, une implantation efficace d'un champ vectoriel pour la modélisation d'une surface 2D dans un volume 3D nécessite l'utilisation d'une structure de données adaptée. L'utilisation d'un *octree* permet une économie de mémoire de plus de 80% comparativement à l'utilisation d'une grille régulière.

Le chapitre suivant explique en détails l'adaptation de la représentation vectorielle présentée dans ce chapitre à la multirésolution. La multirésolution permet d'économiser davantage de mémoire lors de la reconstruction de surface contenant des régions lisses et des détails localisés en permettant à des noeuds feuilles de niveaux de résolution différents d'encoder de l'information sur la surface.

Chapitre 3

Représentation multirésolution dans un champ vectoriel

3.1 Introduction

Le chapitre précédent a présenté la reconstruction de surfaces 3D à l'aide d'une représentation volumétrique basée sur les champs vectoriels. Il s'avère que cette représentation est très bien adaptée à la modélisation interactive. Son avantage principal est de permettre l'ajout d'un nouveau point au modèle en un nombre constant d'instructions, assurant ainsi l'interactivité du système tout au long de la séance d'acquisition. Cela est rendu possible grâce à l'obtention du point le plus près sur la surface en temps constant. Un autre avantage de cette représentation réside en sa capacité de *compression naturelle* imposée par la grille. À l'étape de reconstruction, les données brutes sont combinées en chaque point de la grille dans la matrice de covariance et, par après, détruites. Cette façon de faire permet donc au système de garder sa réactivité peu importe la durée de la séance d'acquisition. Ces avantages indéniables de la représentation vectorielle pour la modélisation interactive ont été obtenus au prix d'une grande consommation de mémoire. C'est pour cette raison que présentement, l'utilisation de cette représentation est limitée à la modélisation d'objets de petite taille ou à faible résolution.

Pour surmonter cette limitation, ce chapitre propose l'utilisation d'une représentation multirésolution inspirée du champ vectoriel [30] comme composante centrale d'un système de modélisation interactive. Cette représentation doit permettre la reconstruction de la surface à différents niveaux de résolution tout en fournissant une *représentation multirésolution cohérente* de l'objet numérisé lors de l'*interprétation des données.*

Des définitions approfondies d'une *représentation multirésolution cohérente* et du principe d'*interprétation des données* seront données dans la suite du chapitre, tout juste après avoir présenté les avantages d'une représentation multirésolution au niveau de la gestion des ressources disponibles ainsi que la structure de base d'une telle représentation. Par après, les notions introduites seront validées expérimentalement. Le chapitre se termine par une explication détaillée du nouvel algorithme de reconstruction multirésolution.

3.2 Avantages de la multirésolution

Une représentation multirésolution utilisée pour la modélisation interactive présente des avantages majeurs au niveau de la gestion des ressources mémoire et du temps nécessaire à la modélisation d'un objet. La représentation multirésolution permet de reconstruire l'objet localement à différents niveaux de résolution. Nous pouvons voir la modélisation multirésolution comme une *cohabitation des niveaux de résolution* dans une même représentation.

Le premier avantage de la multirésolution est de permettre au système de reconstruire le modèle d'un objet à partir de différentes densités de points. Cela est possible grâce à la variation de la taille de l'enveloppe de reconstruction entre les niveaux de résolution.

Comme il a été expliqué au chapitre précédent, l'enveloppe de reconstruction est proportionnelle à la taille des *voxels* du niveau de résolution courant. Un faible niveau de résolution résultant en de gros *voxels* utilise une grande enveloppe de reconstruction. Cette grande enveloppe permet à une région de l'objet balayée rapidement d'être reconstruite adéquatement malgré une faible densité de points. À l'opposé, un niveau de résolution élevé résultant en de petits *voxels* utilise une enveloppe de reconstruction plus petite permettant la reconstruction fidèle des détails de l'objet aux endroits où la densité de points est suffisante. Notons que cette flexibilité permet d'économiser mémoire et temps lors de l'acquisition grossière du contexte. Il est maintenant possible de balayer rapidement le contexte et d'en obtenir une représentation grossière utilisant peu de mémoire tout en laissant la possibilité à l'utilisateur de prendre plus de temps pour modéliser finement les régions d'intérêt.

Le second avantage de la représentation multirésolution est la possibilité de compresser des zones à faible courbure qui ont été reconstruites à un niveau de résolution trop élevé. Dans un champ vectoriel, nous avons vu que chaque point de la grille à

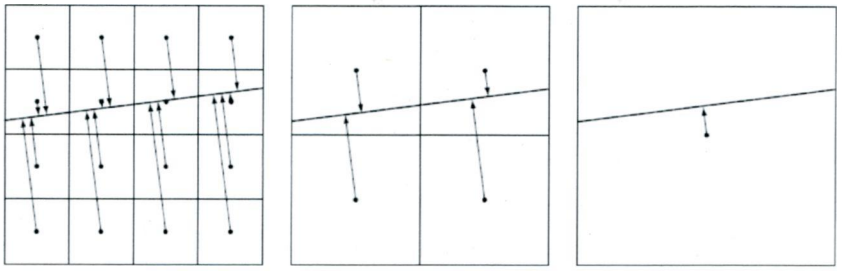
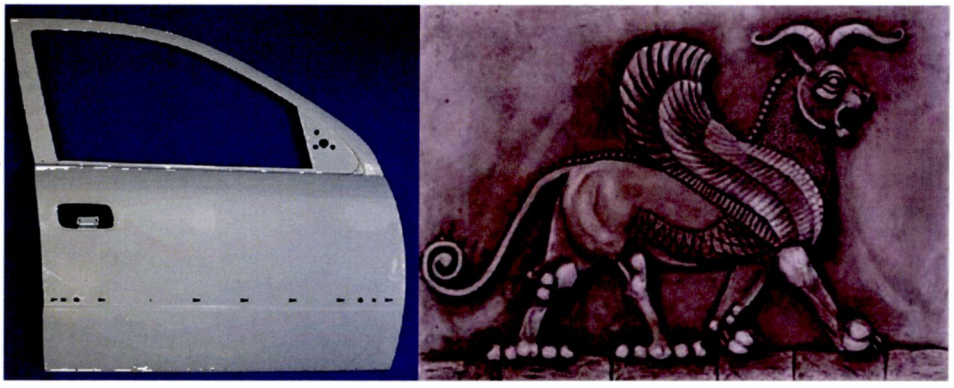


FIG. 3.1 – Redondance des données dans un champ vectoriel pour une zone plane. a) Reconstruction d'une zone plane à un niveau de résolution fin. b) La même surface peut être encodée sans perte d'information au niveau de résolution précédent. c) À la limite, la zone plane peut être encodée par un seul point de la grille tout en décrivant bien la surface. Tirée de [30].

proximité de la surface encode la direction et la distance du point le plus près sur cette dernière. Cela signifie que chaque point de la grille représente une petite section de la surface de l'objet. Comme le montre la figure 3.1, cette approche entraîne de fortes redondances de données pour les zones planes. Dans cet exemple, il serait possible d'encoder le groupe de points de la grille haute résolution de la figure 3.1 a) par les points de la grille du niveau grossier précédent tel que montré à la figure 3.1 b). La figure 3.1 c) montre qu'il serait même possible de pousser encore plus loin la compression en encodant la totalité de la zone plane par un unique point de la grille encore plus grossier. Pour exploiter l'économie de mémoire résultant de cette substitution, il faut absolument que la représentation vectorielle puisse accommoder différents niveaux de résolution.

Beaucoup d'objets présentent une zone de détail localisée et un contexte à courbure faible pouvant bénéficier d'une représentation multirésolution. Tel que montré à la figure 3.2 a), une porte de voiture modélisée à des fins de contrôle de qualité est un bon exemple. La région d'intérêt correspond à la poignée de la porte, qui doit être reconstruite à un haut niveau de résolution. Le contexte correspond au reste de la porte, qui lui est généralement à faible courbure. Un autre exemple présenté à la figure 3.2 b) montre un bas relief entouré d'un contexte plan. Ce type d'objet est fréquent dans les domaines de la muséologie et de la conservation du patrimoine culturel.

Les explications précédentes montrent que l'utilisation de la multirésolution dans un contexte de modélisation interactive présente des avantages importants. La multirésolution permet de reconstruire une surface à partir de différentes densités locales de points et permet également la compression des régions à faible courbure. La section



(a) (b)

FIG. 3.2 – Exemples d'objets comprenant un contexte à faible courbure entourant une zone contenant beaucoup de détails. a) Porte de voiture en cours de production. Tirée de [40]. b) Bas relief représentant un griffon. Tirée de [41].

suivante aborde la représentation multirésolution proposée en mettant l'emphase sur les mécanismes impliqués au niveau de la structure de données. Au chapitre 4, nous examinerons un algorithme de compression temps réel permettant de déterminer automatiquement le niveau de résolution local auquel doit être reconstruit l'objet pour optimiser l'utilisation des ressources disponibles.

3.3 Champ vectoriel multirésolution

Permettre la modélisation rapide du contexte en utilisant peu de mémoire requiert de contrôler le niveau de résolution pour être en mesure de reconstruire une surface à partir de différentes densités de points. La structure multirésolution doit donc s'adapter localement pour permettre la fusion et la reconstruction des données brutes. Il s'avère qu'un *octree,* présenté au chapitre 2, est une structure de données parfaitement adaptée à la tâche. Précédemment, l'*octree* a été utilisé pour éviter d'allouer une grille complète pour encoder le champ vectoriel et ainsi économiser de la mémoire. Seul le niveau de résolution le plus fin contenait de l'information concernant la surface. Les niveaux intermédiaires n'étaient présents que pour assurer la cohérence de la structure de données. Dorénavant, l'*octree* sera utilisé pour permettre la reconstruction du champ vectoriel à différents niveaux de résolution.

Tel que montré à la figure 3.3 a), l'*octree* peut être vu comme une arborescence tridimensionnelle dont la racine correspond au volume lui-même avec un point de la grille situé en son centre. Un *octree* contenant uniquement la racine est un *octree* de niveau

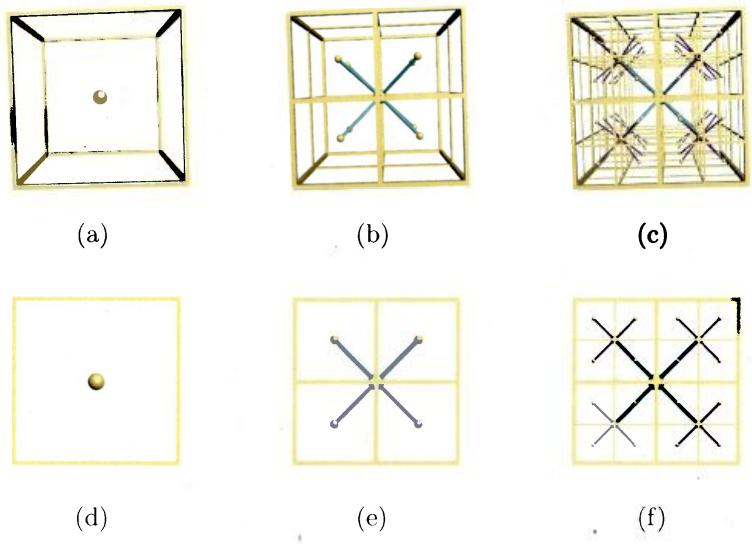


FIG. 3.3 – Emplacement des noeuds de l'*octree* pour différents niveaux de résolution. La ligne du haut présente une vue en perspective de l'*octree*. La ligne du bas présente une vue simplifiée en 2D. a) et d) *Octree* du niveau 0 (la racine). b) et e) *Octree* du niveau 1 contenant 8 noeuds. c) et f) *Octree* du niveau 2 contenant 64 noeuds.

0. À chaque niveau supplémentaire, le volume entourant un point de grille est subdivisé en huit sous-octants de même taille. Les figure 3.3 b) et c) montrent respectivement un *octree* complet de niveau 1 ($8^1 = 8$ *voxels*) et 2 ($8^2 = 64$ *voxels*). Tel que montré au bas de la figure 3.3, une représentation simplifiée 2D sera utilisée pour représenter l'*octree* dans le reste du document dans le but de simplifier les démonstrations. À noter qu'il n'est plus tout à fait juste d'utiliser l'expression *point de la grille* pour désigner les positions du volume contenant de l'information sur la surface. Puisque l'*octree* est en fait une structure de données en arbre, l'expression *noeud* sera dorénavant utilisée en remplacement de *point de la grille*.

Grâce à sa structure en arbre, un *octree* peut facilement permettre la sauvegarde des données à différents niveaux de résolution. L'idée maîtresse au coeur du champ vectoriel multirésolution est de permettre aux noeuds de l'*octree* à des niveaux intermédiaires de contenir de l'information sur la surface et non pas uniquement d'encoder la surface au niveau le plus fin. La figure 3.4 montre un exemple d'une surface encodée à trois niveaux de résolution différents. Il est important de remarquer que la taille des *voxels* entre les niveaux de résolution adjacents est divisée par 2 et que les noeuds d'un niveau enfant sont positionnés autour du noeud parent à un distance de $\pm\frac{\Delta parent}{4}$ selon chacun des axes.

L'enveloppe de reconstruction dans un champ vectoriel multirésolution est gérée

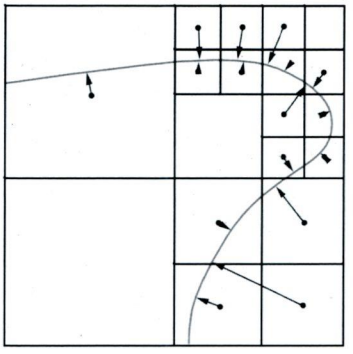


FIG. 3.4 – Coupe d'un champ vectoriel multirésolution contenant une surface encodée à trois niveaux de résolution différents.

d'une façon très similaire à celle utilisée pour le champ vectoriel monorésolution. L'idée est toujours de fixer une constante ε donnant en *voxels* la taille de l'enveloppe de reconstruction. Il est important de noter que tous les niveaux de résolution utilisent la même valeur de ε. Étant donné que la taille des *voxels* est divisée par 2 entre chaque niveau adjacent, la taille des enveloppes de reconstruction des noeuds est également divisée par 2 à chaque incrément de niveau. Cela implique qu'un noeud au niveau de résolution $n+1$, ayant une enveloppe de reconstruction de taille $\varepsilon\Delta_{n+1}$, encode une section de surface avec un rayon deux fois plus petit qu'un noeud au niveau n, qui lui a une enveloppe de reconstruction de taille $\varepsilon\Delta_n = 2\varepsilon\Delta_{n+1}$.

C'est la présence de *voxels* de différentes tailles à l'intérieur d'une même représentation qui permet au champ vectoriel multirésolution de reconstruire une surface à partir de différentes densités de points. En présence de données bruitées, un noeud passera le *test de validité de surface* (section 2.4.3) seulement si un nombre suffisant de données a été recueilli dans son enveloppe. Dans le cas contraire, le petit nombre de données bruitées fera échouer le test de validité. On note ici que l'enveloppe du champ vectoriel fait office de filtre passe bas sur les données. Étant donné que la taille de l'enveloppe est proportionnelle à la taille des *voxels*, un faible niveau de résolution peut encoder une surface valide avec une densité de points beaucoup plus faible qu'un niveau de résolution fin à cause de la grande taille de son enveloppe de reconstruction. Pour un niveau grossier, même si la densité est faible, un plus grand nombre de données contribueront à la reconstruction du noeud. Si la surface était reconstruite à haute résolution à partir d'une faible densité de données bruitées, un nombre insuffisant de données seraient mesurées dans l'enveloppe de reconstruction et la représentation à ce noeud risquerait de ne pas passer le *test de validité de surface.*

Il est important de noter que tous les niveaux de résolution à l'intérieur de la représentation sont indépendants. Cela signifie qu'il est possible qu'une même section de

surface soit encodée à différents niveaux de résolution. La section suivante montre comment une reconstruction progressive des niveaux de résolution assure une *représentation cohérente* de la surface.

3.4 Interprétation et cohérence du champ vectoriel

Nous venons de voir que la représentation vectorielle multirésolution est basée sur un *octree* et permet la reconstruction locale de la surface à différents niveaux de résolution. La question qui se pose maintenant est comment reconstruire une *surface multirésolution cohérente* ? Cette question fait intervenir la notion de *cohérence de la surface* qui est fortement liée à la notion d'*interprétation* du champ vectoriel multirésolution.

3.4.1 Interprétation du champ vectoriel

Tout d'abord, définissons ce qu'est l'*interprétation* du champ vectoriel. L'*interprétation* est l'étape à laquelle les données encodées dans le champ vectoriel sont lues pour en extraire de l'information sur la surface reconstruite. L'*interprétation* du champ vectoriel peut prendre différentes formes.

Une forme d'*interprétation* très fréquente, qui sera abordée en détails au chapitre 5, concerne l'affichage de la surface encodée. L'affichage de la surface à l'aide d'une approche par lancer de rayons demande au système de déterminer la position et l'orientation de la surface encodée dans la représentation. Détecter la présence de la surface le long d'un rayon est une forme d'*interprétation*. C'est à cette étape que les différentes sections de surface discontinues reconstruites dans le champ vectoriel fournissent une représentation de la surface de l'objet pouvant être appréciée par l'utilisateur. Une approche de rendu par projection de pastilles est également une forme d'*interprétation* ayant pour but l'affichage de la surface en cours de reconstruction.

Les approches par lancer de rayons ou par projection de pastilles interprètent le champ vectoriel en fournissant une image d'un point de vue de la surface encodée. Cependant, cette forme d'*interprétation* ne permet pas de prendre des mesures de la surface de l'objet ou de l'utiliser à des fins d'évaluation de la qualité ou de prototypage. Un moyen efficace de pouvoir interagir aisément avec la surface encodée est d'en extraire une triangulation. La génération d'une triangulation à partir d'un champ vectoriel est

une autre forme d'*interprétation*. Des algorithmes tels que les *Marching Cubes [21]* et le *Dual Contouring* [15] interprètent les données encodées dans le champ vectoriel dans le but de générer une liste de triangles qui pourra être utilisée dans une foule d'applications. Les détails de l'implantation du *Dual Contouring* peuvent être trouvés aux références [27] et [9]. La génération d'une triangulation à partir d'un champ vectoriel multirésolution ne sera pas expliquée plus en détails, car c'est une problèmatique de taille qui n'a pas été abordée dans ces travaux.

Nous verrons également au chapitre 4 que l'algorithme de compression en temps réel est une forme d'*interprétation* des données contenues dans le champ vectoriel. Durant l'ajout de données brutes, l'algorithme de compression interprète des sections locales de la surface et détermine si la compression peut avoir lieu ou pas.

Ces exemples simples cachent un défi de taille : fournir une *interprétation cohérente* de la surface. Cela ne sera possible que si la représentation encodée dans le champ vectoriel est également *cohérente*.

3.4.2 Cohérence du champ vectoriel

Pour être en mesure de fournir une *interprétation multirésolution cohérente* de la surface, il faut que les données encodées dans le champ vectoriel soient *cohérentes*. Comme la *cohérence du champ vectoriel multirésolution* est un concept plutôt abstrait, nous allons l'aborder dans un premier temps de manière informelle. Suivra ensuite une définition mathématique de la cohérence qui sera appuyée par un exemple concret.

Qu'est-ce qu'un champ vectoriel multirésolution *cohérent* ? Une réponse simple à cette question pourrait être qu'un champ vectoriel multirésolution *cohérent* fournit, lors de l'*interprétation* par le module de visualisation, un rendu sans défaut de la surface encodée. Un rendu sans défaut ne présenterait pas de trous à des endroits non voulus et afficherait le niveau de résolution le plus fin disponible en chaque endroit de la surface. Un autre exemple de champ vectoriel multirésolution *cohérent* serait un champ qui, lors de l'extraction d'une triangulation, ne génèrerait pas de surfaces doubles dues à une incohérence entre les différents niveaux de résolution pour une même section de surface.

La figure 3.5 montre un exemple de la reconstruction du champ vectoriel à différents niveaux de résolution. Il est très fréquent qu'une section de surface soit reconstruite à plus d'un niveau de résolution. Un exemple d'une telle situation est la reconstruction du contexte à un niveau grossier suivi de la modélisation à un niveau plus fin d'une région d'intérêt. Ainsi, la région d'intérêt possède une représentation grossière et une

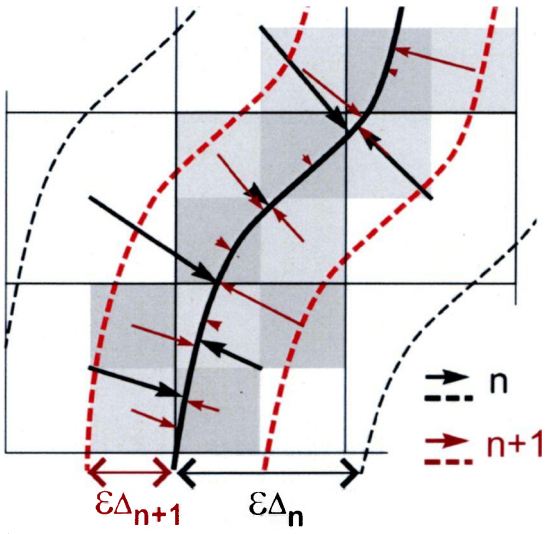


FIG. 3.5 – Cohabitation de deux niveaux de résolution adjacents. Notez la taille de l'enveloppe du niveau $n+1$ qui est deux fois plus petite que celle du niveau n.

représentation plus fine, toutes deux encodées dans le même champ vectoriel. Cette situation correspond à la cohabitation de deux niveaux de résolution représentant la même surface.

Plus précisément, la figure 3.5 présente la cohabitation de deux niveaux de résolution adjacents. Le niveau le plus grossier n correspond aux noeuds tracés en noir et ayant une taille d'enveloppe de $\varepsilon\Delta_n$. Le niveau le plus fin $n+1$ correspond aux noeuds tracés en rouge sur fond gris et ayant une taille d'enveloppe de $\varepsilon\Delta_{n+1} = \frac{\varepsilon\Delta_n}{2}$. Le défi de l'*interprétation d'une surface cohérente* est donc d'extraire la bonne surface du champ vectoriel. Pour ce faire, les algorithmes d'*interprétation* doivent suivre la règle suivante :

Règle d'interprétation des données : Pour tout point du volume, les algorithmes d'interprétation doivent utiliser le niveau de résolution valide le *plus fin* disponible. Les autres niveaux plus grossiers présents doivent être ignorés lors de l'interprétation de la surface même s'ils contiennent des données valides.

Cette règle d'interprétation implique que les niveaux de résolution fins ont priorité sur les niveaux de résolution grossiers. Cela peut être vu comme si les niveaux de résolution fins désactivent les données des niveaux plus grossiers situées dans leur entourage. C'est cette désactivation des niveaux de résolution grossiers qui permet de garder la cohérence de la représentation durant la reconstruction.

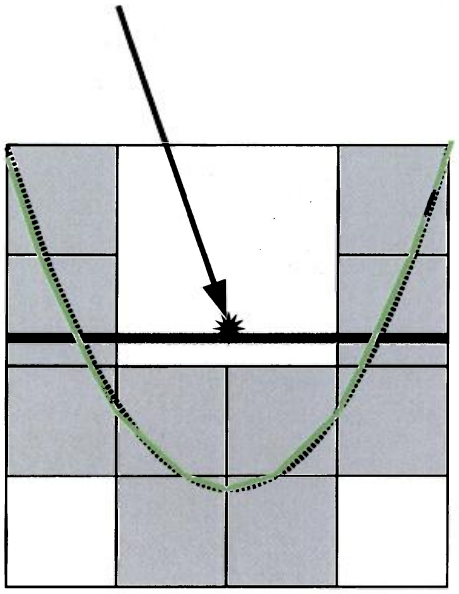


FIG. 3.6 – Problème d'interprétation de l'algorithme basé sur l'approche par lancer de rayons. Le rayon rencontre en premier la surface encodée au niveau grossier n (affichée en pointillés rouges) même si cette section de surface est représentée au niveau plus fin $n+2$ par une surface valide (affichée en traits verts).

Les problèmes surviennent lorsque, pour une même section de surface, les données encodées au niveau le plus fin disponible ne recouvrent pas les données encodées aux niveaux plus grossiers. Cela signifie qu'une section de surface est représentée localement à deux niveaux de résolution différents par des données valides. Les conséquences d'une telle *incohérence* dans le champ vectoriel multirésolution dépendent de l'algorithme d'interprétation utilisé. Lors de l'extraction d'une triangulation, une surface double serait présente à cet endroit. Tel que montré à la figure 3.6, pour une interprétation à partir d'une approche par lancer de rayons, cette situation génèrerait un rendu dans lequel une section du niveau grossier cacherait la surface encodée au niveau plus fin. Analysons maintenant pourquoi cet effet désagréable est présent et comment faire pour l'éviter.

Revenons plus en détails sur la figure 3.6. Pour cet exemple, l'enveloppe de reconstruction est fixée à la taille minimale permise, soit la moitié de la diagonale d'un *voxel*. La représentation de la surface encodée au niveau grossier n, soit le plus grand *voxel* visible dans la figure, a été obtenue à partir d'un très grand voisinage qui fait en sorte que l'erreur maximale entre la vraie surface (pointillés noirs) et la représentation via un plan tangent est élevée. Cette erreur maximale est même supérieure à la taille de l'enveloppe de reconstruction du niveau $n+2$ affiché en gris. Cela fait en sorte que pour certaines zones, dans ce cas-ci, le creux du sinus, la surface se dédouble. Il y a

une surface encodée au niveau n, présentée en rouge pointillé, et une autre encodée au niveau $n+2$, présentée par plusieurs traits verts. Pour cette région du volume, le champ encode donc deux surfaces qui, dans les faits, devraient être une seule et même surface.

La première règle devant être respectée lors de la reconstruction du champ vectoriel multirésolution pour éviter ce problème d'incohérence est la suivante :

Continuité des niveaux de résolution : Pour que la cohérence d'un champ vectoriel multirésolution soit conservée, il faut que tous les niveaux de résolution entre le niveau grossier et le niveau le plus fin localement disponible soient reconstruits et valides.

Intuitivement, la continuité des niveaux de résolution semble effectivement être une solution pour conserver la cohérence de la représentation multirésolution. L'idée est de désactiver progressivement les données encodées par un niveau grossier n à l'aide de noeuds valides appartenant au niveau adjacent $n+1$. Mathématiquement, la continuité des niveaux de résolution assurera la cohérence du champ multirésolution si la condition suivante est respectée :

$$Surface_n \subset Enveloppe_{n+1} \tag{3.1}$$

Cette condition implique que localement, la surface encodée au niveau n, c'est-à-dire le plan tangent encodé à ce niveau de résolution, doit être recouverte par des noeuds valides appartenant au niveau $n+1$. Il ne faut pas confondre la surface du niveau n avec le volume défini par le noeud du niveau n. Lorsque la taille de l'enveloppe de reconstruction est supérieure à un *voxel,* il n'est pas rare que le plan tangent encodé soit à l'extérieur du volume défini par le noeud.

Cette condition n'est évidemment pas respectée peu importe les conditions de modélisation. De fortes erreurs de mise en registre entre l'acquisition des données du contexte et celles recueillies par après pour les zones de détails ou des données en provenance d'un capteur très bruité peuvent faire en sorte que la condition 3.1 ne soit pas toujours respectée. Cela aura pour conséquence la présence d'une double surface dans le champ vectoriel générant les problèmes d'interprétation précédemment expliqués. Néanmoins, l'expression 3.1 sera respectée pour un noeud p_v du niveau n si l'inéquation suivante est vraie pour tous les points p_i mesurés dans l'enveloppe de reconstruction de ce noeud :

$$\Gamma_{max} < \varepsilon \frac{\Delta_n}{2} = \varepsilon \Delta_{n+1} \tag{3.2}$$

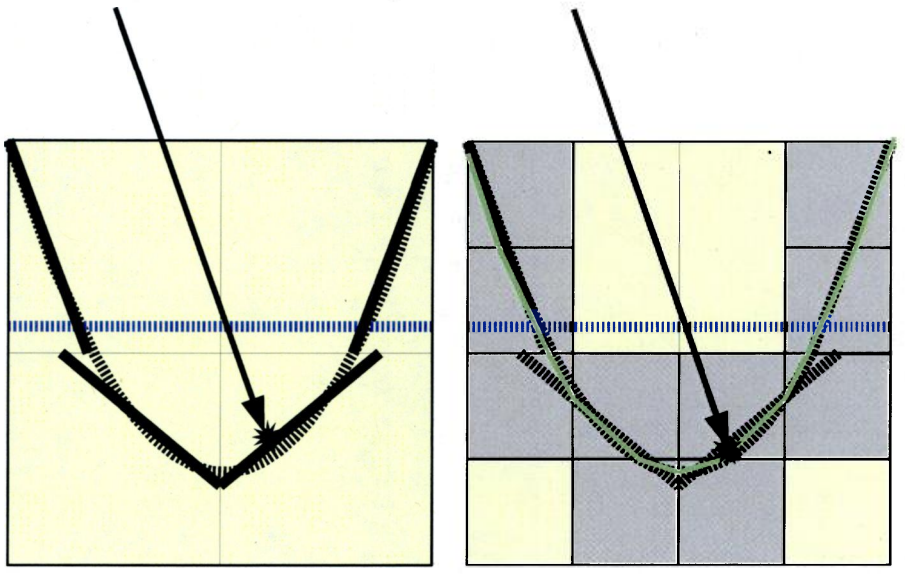
où

$$\Gamma_{max} = argmax_i(dist(p_i, F_n(p_v)). \tag{3.3}$$

L'opérateur $dist(p_i, F_n(p_v))$ retourne la distance entre un point p_i et la représentation de la surface encodée dans le champ vectoriel au niveau n pour le noeud p_v. La variable Γ_{max} correspond en fait à la distance maximale entre tous les points mesurés dans l'enveloppe de reconstruction d'un noeud du niveau n et le plan tangent qu'il encode. Cela peut être également vu comme l'erreur d'encodage maximale.

L'inéquation 3.2 montre que si Γ_{max} est inférieure à la taille de l'enveloppe de reconstruction du niveau $n+1$, alors le plan tangent du niveau n devrait pouvoir être recouvert par des noeuds valides du niveau de résolution suivant. Cela assure la possibilité de désactiver la représentation grossière du niveau n par une représentation du niveau $n+1$ lorsqu'elle sera reconstruite. Si la distance maximale Γ_{max} est supérieure à $\varepsilon\Delta_{n+1}$, alors il se peut que la surface encodée au niveau n ne puisse pas être complètement désactivée par des noeuds du niveau $n+1$. Cette réflexion implique que les noeuds grossiers ne respectant pas l'inéquation 3.2 doivent être considérés invalides et ne doivent pas intervenir lors de l'étape d'interprétation du champ vectoriel.

Revenons maintenant sur le problème présenté à la figure 3.6. Le même scénario est montré à la figure 3.7, mais en appliquant le principe de continuité des niveaux de résolution entre le niveau grossier n et le niveau fin $n+2$. La figure 3.7 a) montre l'état de la représentation suite à la reconstruction du niveau $n+1$. Les noeuds présentant des données valides à ce niveau recouvrent complètement le plan tangent encodé au niveau n. Cela a pour conséquence de désactiver les sections de surface encodées au niveau grossier, présentées en pointillés bleus, car elles sont complètement recouvertes de noeuds valides appartenant à un niveau supérieur $(n+1)$. La continuité des niveaux assure donc une désactivation progressive des données contenues dans les niveaux de résolution grossiers autour de la surface de l'objet. Si à ce moment un rendu de la surface aurait été nécessaire, une approche par lancer de rayons aurait pu afficher la surface au niveau le plus fin disponible, soit $n+1$. La figure 3.7 b) montre que, grâce à la reconstruction du niveau $n+1$, il est maintenant possible de reconstruire le niveau $n+2$ de façon cohérente, sans générer de doubles surfaces. Pour cet exemple, c'est effectivement la surface reconstruite au niveau le plus fin, représentée par des traits verts, qui serait utilisée lors de l'interprétation des données encodées dans le champ.



(a) **(b)**

FIG. 3.7 – Explication du principe de continuité des niveaux de résolution. a) La présence de noeuds valides au niveau $n+1$ désactive le plan tangent encodé au niveau n (pointillés bleus). Lors de l'interprétation, c'est la surface au niveau $n+1$ qui sera affichée. b) Comme il y a continuité des niveaux de résolution, les noeuds valides au niveau $n+2$ sont en mesure de désactiver les données encodées au niveau $n+1$. Dans cette situation, la représentation du niveau $n+2$ (en vert) serait affichée.

Estimation de Γ_{max}

Comme le système ne stocke pas les données brutes, mais seulement une approximation de leur répartition dans l'espace grâce au centre de masse et aux données contenues dans la matrice de covariance, il n'est pas possible de calculer exactement l'erreur d'encodage Γ_{max}. Le seul élément sur lequel on peut se fier pour déterminer l'erreur de représentation est la plus petite valeur propre de la matrice de covariance. Elle est associée au vecteur propre qui est perpendiculaire au plan tangent. Elle donne une idée de la répartition du nuage de points situé à l'intérieur de l'enveloppe de reconstruction selon la normale au plan. En pratique, pour pallier au bruit des données brutes, aux erreurs de mise en registre ainsi qu'au phénomène de discrétisation des points de la grille, le seuil utilisé par le test de validité présenté au chapitre 2 est beaucoup plus sévère que $\frac{\varepsilon\Delta}{2}$. Cela signifie qu'il n'est pas nécessaire d'ajouter un nouveau test de validité lors de la mise à jour du champ multirésolution, car le test actuel est déjà plus sévère que la borne théorique maximale ne devant pas être dépassée. Cependant, comme ce test de validité se base sur la plus petite valeur propre de la matrice de covariance qui correspond à une approximation de la répartition du nuage de points, il n'y a aucune garantie que la condition soit toujours respectée. Cependant, le système a été validé expérimentalement par la modélisation multirésolution avec succès de plusieurs objets montrant que cette condition est généralement respectée.

Surcharge due à la continuité des niveaux de résolution

Maintenir les niveaux intermédiaires entraîne une consommation accrue de mémoire, car dorénavant, les noeuds de l'*octree* situés entre le niveau N et le niveau N+P doivent contenir la structure de données permettant d'encoder un plan tangent à la surface en cours de reconstruction. Cependant, cette surcharge n'est pas très élevée car le nombre de noeuds contenus entre le niveau N et N+P-1 ne correspond qu'à 33% du nombre de noeuds contenus au niveau fin N+P. Pour obtenir ce résultat, il faut voir le nombre de noeuds utilisés pour encoder la surface au niveau N+P comme étant εA, où A correspond à l'aire de la surface et ε correspond à l'épaisseur de l'enveloppe, en *voxels*. Au niveau N+P-1, le nombre de noeuds nécessaires tombe à $\frac{\varepsilon A}{4}$. La densité de noeuds selon la surface a diminué par 4, mais l'épaisseur de l'enveloppe est toujours de ε noeuds. Le niveau N+P-2 utilise $\frac{\varepsilon A}{16}$ noeuds et ainsi de suite. Cette progression de $\frac{\frac{\varepsilon A}{4}+\frac{\varepsilon A}{16}+\frac{\varepsilon A}{64},\cdots}{\varepsilon A}$ tend vers 33%.

3.5 Expérimentations

3.5.1 Implications pratiques

La discussion précédente laisse supposer qu'il serait avantageux de pouvoir reconstruire la surface du modèle à n'importe quel niveau de résolution de l'*octree*. Cependant, cette idée est fausse en pratique car il est préférable de limiter les niveaux de résolution auxquels il est permis d'encoder la surface. Il n'est pas utile de reconstruire les niveaux de résolution très grossiers inférieurs à 6 ou 7, c'est-à-dire comportant environ 64 à 128 noeuds par côté de l'*octree*. En deça de cette résolution, très peu de noeuds passeraient le test de validité de la surface pour la plupart des objets. Cela est dû au fait que l'enveloppe de reconstruction pour de tels niveaux possède un très grand support et l'approximation des données brutes par un plan engendrerait une erreur trop élevée. Donc, par souci d'économie de ressources mémoire et processeur, seuls les niveaux supérieurs ou égaux à un niveau minimum N sont reconstruits.

Une fois que le système a la possibilité de reconstruire la surface à différents niveaux de résolution, la problématique principale est de déterminer quel est localement le niveau de résolution optimal. Cette unique question fera l'objet du prochain chapitre. Cependant, il est possible de valider la reconstruction multirésolution avec un système plus simple qui nécessite l'aide de l'utilisateur pour déterminer le niveau de résolution auquel la surface doit être reconstruite.

Dans un tel scénario, une acquisition typique débute généralement par l'acquisition du contexte. Pour ce faire, l'utilisateur active le niveau de résolution grossier N et balaye rapidement le contexte de l'objet. Lorsqu'il est satisfait du résultat qu'il voit à l'écran et qu'il juge que le contexte a été reconstruit correctement, il active un niveau de résolution plus élevé pour faire l'acquisition des zones de détails. Généralement, ce niveau fin nommé N+P est 3 à 4 niveaux supérieurs au niveau de base. Pour un niveau grossier $N = 7$, cela donne un volume ayant aux environs de $2^{(7+4)} = 2048$ noeuds par côté de l'*octree*. Lorsque le niveau de résolution fin est activé, l'utilisateur peut balayer lentement les zones d'intérêt et recueillir une forte densité de points pour rendre la surface valide malgré la petite taille des enveloppes de reconstruction.

3.5.2 Densité VS résolution

L'expérience suivante met en pratique les notions précédemment expliquées. Le but de cette expérience est d'évaluer le comportement de l'algorithme de reconstruction en fonction de la densité de points fournie et du niveau de résolution actif, choisi manuellement par l'utilisateur. Les résultats de cette expérience sont montrés à la figure 3.8.

La figure 3.8 a) montre le résultat de la reconstruction de la surface à un faible niveau de résolution $N = 7$ suite au balayage rapide du contexte. Cette figure montre qu'une faible densité de points est suffisante pour reconstruire adéquatement le contexte à un niveau grossier. De cette manière, il est possible pour un utilisateur de reconstruire un contexte complet de l'objet en un court laps de temps et en consommant peu de mémoire. La figure 3.8 b) montre la reconstruction de la même surface au même niveau de résolution, mais cette fois avec une densité de points élevée. Le point important à noter ici est le faible gain au niveau de la qualité de reconstruction de la surface entre une acquisition à faible densité et une acquisition à forte densité lorsque le niveau de résolution est faible. Cela signifie que pour l'acquisition du contexte, une faible densité de points est suffisante pour obtenir un résultat satisfaisant.

La seconde étape de l'expérience consiste à reconstruire la surface à un niveau de résolution plus élevé, c'est à dire $N + P = 9$, mais en réutilisant les mêmes ensembles de points. La figure 3.8 c) montre donc la reconstruction de la surface à un niveau de résolution élevé à partir d'une faible densité de points. La présence de trous dans la surface montre qu'une faible densité de points de paire avec de petites enveloppes de reconstruction ne permet pas une reconstruction cohérente de la surface. Lorsque la densité de points augmente, tel que montré à la figure 3.8 d), la surface est reconstruite à un haut niveau de résolution et il est possible de distinguer tous les détails de l'objet. Notez que le pourtour de la zone de détail n'a pas été balayé suffisamment densément pour permettre une reconstruction adéquate de la surface.

La figure 3.8 e) montre un modèle hybride exploitant les avantages de chacune des étapes précédentes. Tout d'abord, il y a remplissage des trous dans le pourtour de l'image suite à l'acquisition rapide du contexte qui a été reconstruit à un faible niveau de résolution. Par après, l'utilisateur a activé un niveau de résolution plus élevé et a balayé de nouveau la zone d'intérêt pour la reconstruire à un haut niveau de résolution.

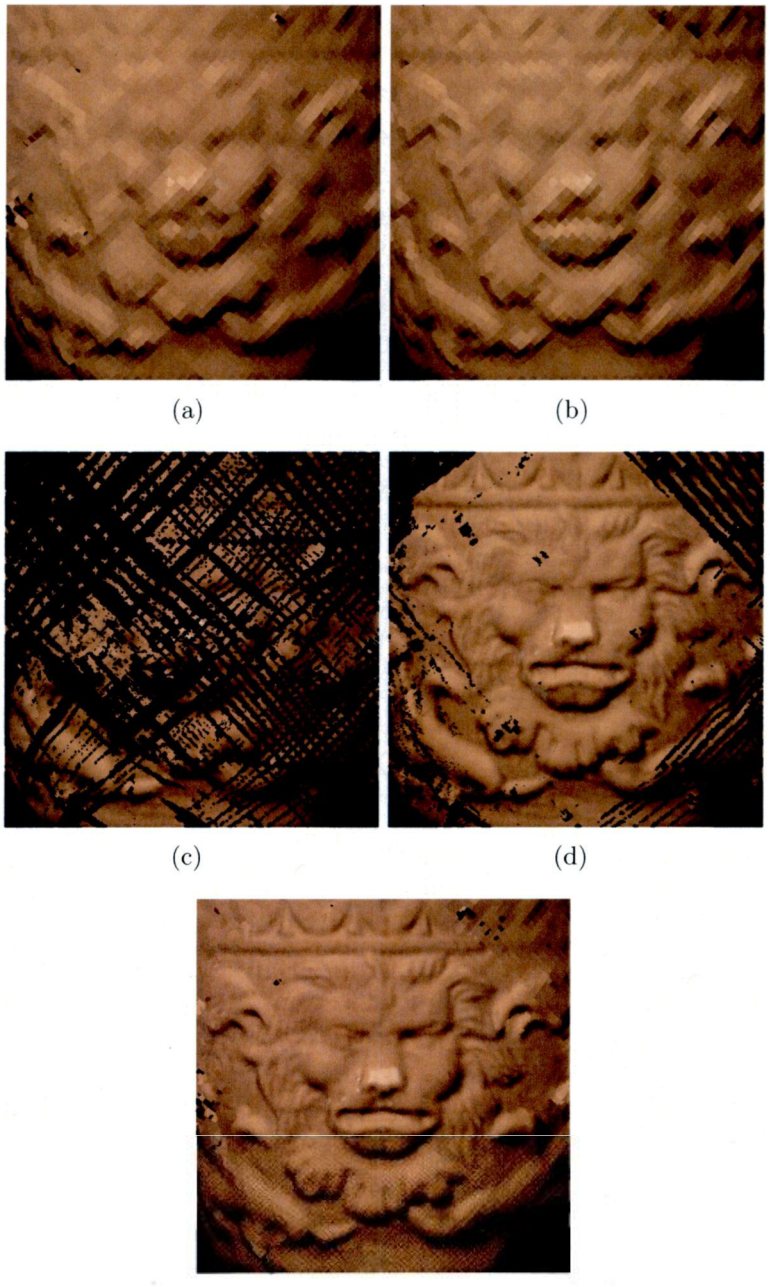


FIG. 3.8 – Effet de la densité des données brutes sur la reconstruction du champ vectoriel. a) Reconstruction de la surface à un faible niveau de résolution (N=7) pour une faible densité de points. b) Faible niveau de résolution (N=7) et forte densité de points. c) Niveau de résolution élevé (N+P=9) et faible densité de points. d) Niveau de résolution élevé (N+P=9) et forte densité de points. e) Résultat final utilisant un faible niveau de résolution pour la modélisation du contexte et un niveau de résolution élevé pour la zone contenant des détails.

3.6 Algorithme de reconstruction multirésolution

Voyons maintenant le mécanisme permettant de reconstruire progressivement les niveaux de résolution intermédiaires, entre le niveau grossier N et le niveau de résolution fin N+P. Lorsqu'un point 3D est ajouté au modèle, il est premièrement ajouté à tous les *voxels* du niveau le plus grossier N faisant partie de sa cellule fondamentale. À ce niveau, la cellule fondamentale correspond à une sphère de rayon $\varepsilon\Delta_N$ centrée sur la position du point 3D ajoutée au champ. Ensuite, le nouveau point 3D est ajouté récursivement à tous les enfants des *voxels* grossiers tant et aussi longtemps que les positions des noeuds enfants tombent dans les cellules fondamentales du point 3D à leurs niveaux respectifs. Pour un noeud du niveau N+1, la cellule fondamentale correspond à une sphère de rayon $\varepsilon\frac{\Delta_N}{2} = \varepsilon\Delta_{N+1}$ centrée sur la position du nouveau point 3D. Pour un noeud du niveau N+2, le rayon de la sphère n'est que de $\frac{\varepsilon\Delta_{N+1}}{2} = \varepsilon\Delta_{N+2}$. L'algorithme 2 explique en détails les étapes suivies pour mettre à jour le champ multirésolution.

Cet algorithme est une modification de celui présenté au chapitre 2. Le tout commence par une triple boucle parcourant la boîte englobante de la cellule fondamentale au niveau le plus grossier. Ensuite, un appel est fait à la fonction *MaJRec(...)* avec comme paramètres un pointeur sur le noeud devant être mis à jour, sa position logique (x, y, z) ainsi que le niveau de résolution du noeud. Cette fonction a pour tâche de mettre à jour le champ vectoriel si la position du noeud fait partie de la cellule fondamentale du point 3D au niveau N. Ensuite, si le niveau N+P n'est pas atteint, elle fait un appel récursif à chacun de ses enfants en calculant la nouvelle coordonnée logique. La coordonnée logique des enfants est calculée à partir de la coordonnée logique du parent et de l'index des enfants, de 0 à 7, écrit en binaire sur 3 bits. Le bit de poids faible correspond à une translation selon l'axe des X, le second bit correspond à une translation selon l'axe des Y et finalement, le bit de poids fort correspond à une translation selon l'axe des Z.

Il est important de noter que dans la représentation volumétrique multirésolution, les modèles reconstruits à différents niveaux de résolution ne sont pas fusionnés. Ils sont complètement indépendants les uns des autres. Cela signifie que le champ multirésolution contient plusieurs représentations de la surface à différents niveaux de résolution. Les niveaux sont reconstruits de manière totalement indépendante, sans réutiliser les données des niveaux inférieurs, pour deux raisons majeures.

Tout d'abord, initialiser les données du niveau $n+1$ avec la représentation du niveau n pourrait entraîner un biais qui ne serait pas possible de corriger, même en ajoutant une infinité de points. Cela est dû au fait que lors de la mise à jour du champ, les données sont constamment ajoutées à la matrice de covariance. Il n'y a aucun mécanisme permettant

$p_i \leftarrow$ Nouveau point ajouté à la représentation
$p_L \leftarrow$ coordonnées logiques du point de la grille p_v le plus près de p_i (voir eq. 2.1)
$N \leftarrow$ niveau de résolution le plus grossier
$P \leftarrow$ nombre de niveaux devant être reconstruits
$\Delta_n \leftarrow$ taille des *voxels* du niveau n
Pour x **de** $p_{L_x} - \lceil\varepsilon\rceil$ **à** $p_{L_x} + \lceil\varepsilon\rceil$ **faire**
  Pour y **de** $p_{L_y} - \lceil\varepsilon\rceil$ **à** $p_{L_y} + \lceil\varepsilon\rceil$ **faire**
    Pour z **de** $p_{L_z} - \lceil\varepsilon\rceil$ **à** $p_{L_z} + \lceil\varepsilon\rceil$ **faire**
      $Noeud_{courant} \leftarrow$ Noeud à la position logique (x,y,z)
      MaJRec($Noeud_{courant}$,x,y,z,N)
    Fin Pour
  Fin Pour
Fin Pour
Fonction MaJRec(Noeud, x, y, z, n) :
  $p_{physique} \leftarrow \Delta_n(x,y,z) + (\frac{\Delta_n}{2}, \frac{\Delta_n}{2}, \frac{\Delta_n}{2})$
  $d_{physique} \leftarrow \|p_{physique} - p_i\|$
  Si $(d_{physique} < \varepsilon\Delta_n)$ **Alors**
    Mettre à jour le point de grille situé à la position logique (x,y,z) et au niveau n
  Fin Si
  Si $(n < N + P)$ **Alors**
    Pour i **de** 1 **à** 8 **faire**
      MaJRec($Noeud \rightarrow enfant_i$, x*2+(i&1! = 0), y*2+(i&2! = 0), z*2+(i&4! = 0), $n+1$)
    Fin Pour
  Fin Si
Fin

Algorithme 2: Algorithme utilisé pour mettre à jour le champ vectoriel multirésolution entre les niveaux N et N+P lors de l'ajout de nouvelles données.

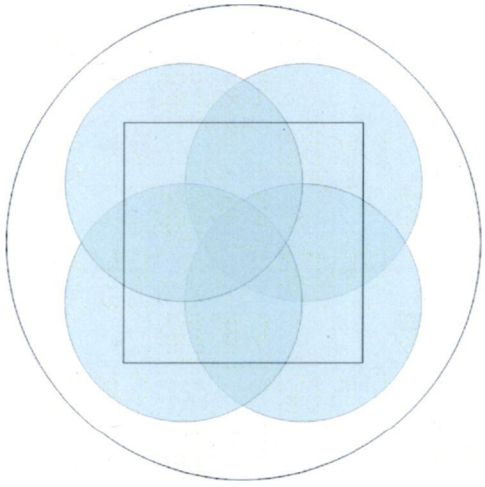


FIG. 3.9 – Différence de couverture des enveloppes de reconstruction entre deux niveaux adjacents (n et $n+1$). L'enveloppe du niveau n correspond au grand cercle noir et les enveloppes du niveau $n+1$ correspondent aux quatre cercles bleus.

d'enlever l'influence des données erronées. Donc, si la matrice de covariance est initialisée avec une représentation comprenant beaucoup d'erreur, cette erreur sera diminuée avec l'ajout de nouveaux points, mais elle ne pourra jamais disparaître complètement.

Tel que montré à la figure 3.9, la seconde raison faisant en sorte que l'initialisation du niveau $n+1$ à partir des données contenues au niveau n n'a pas été retenue est que l'enveloppe du niveau n, montrée par un cercle noir sur la figure, ne couvre pas le même volume que l'intersection des enveloppes de ses enfants du niveau $n+1$, correspondant aux quatre cercles bleus. Cette constatation montre que l'initialisation du niveau enfant avec les données du niveau parent n'entraînerait pas seulement un biais, mais influencerait la reconstruction des noeuds du niveau $n+1$ avec des données mesurées à l'extérieur de leur enveloppe de reconstruction.

Malgré le fait que les conditions assurant la continuité des niveaux de résolution sont respectées par l'algorithme de reconstruction, il se peut que des discontinuités se produisent dans le champ vectoriel multirésolution. À l'étape de reconstruction de la surface, le but n'est pas de cacher ou d'éliminer toutes les discontinuités possibles. Une discontinuité représente généralement une incohérence entre les différents niveaux de résolution. C'est à l'étape de l'interprétation de détecter ce genre d'erreur et de réagir en conséquence. Par exemple, un module de visualisation par lancer de rayons pourrait aviser l'utilisateur d'une discontinuité dans le champ vectoriel en affichant cette région d'une couleur différente. Cependant, l'implantation du module de visualisation présenté

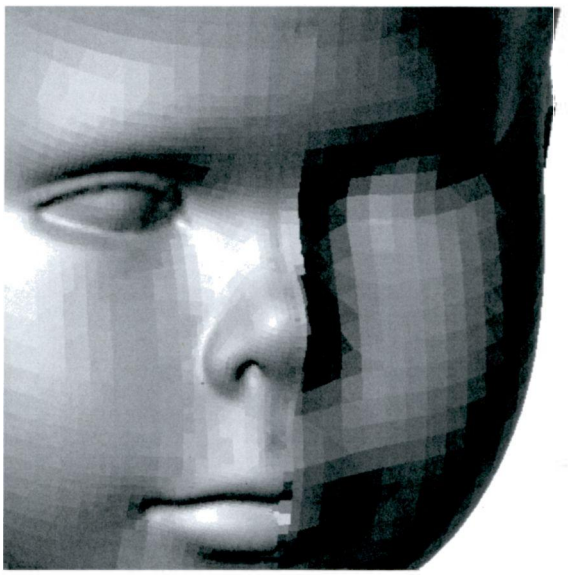


FIG. 3.10 – Simulation montrant les problèmes de discontinuités lorsque les transitions entre les niveaux de résolution ont lieu dans des régions contenant de fortes courbures. Le côté droit de la tête de mannequin a été reconstruit à une résolution maximale de 512 noeuds par côté de l'*octree* et le côté gauche a été reconstruit à une résolution de 64 noeuds par côté de l'*octree*. Il est important de noter que le mécanisme de sélection automatique du niveau de résolution faisant l'objet du prochain chapitre a été utilisé pour générer cette image, d'où la présence de plus d'un niveau de résolution sur le côté droit de la tête du mannequin.

au chapitre 5 ne tente pas d'éliminer les discontinuités présentes dans le champ vectoriel, car ces dernières indiquent qu'il y a un problème au niveau de l'alignement des données ou que certaines régions du modèle ne sont pas reconstruites au niveau de résolution adéquat.

La figure 3.10 montre un exemple de discontinuités lorsque la transition entre les niveaux de résolution a lieu dans des régions contenant de fortes courbures. Pour cette simulation, le côté droit de la tête de mannequin a été reconstruit en permettant un niveau de résolution élevé, soit une résolution de 512 noeuds par côté. Il est à noter que le mécanisme de sélection automatique du niveau de résolution faisant l'objet du prochain chapitre a été utilisé pour générer cette image, d'où la présence de plus d'un niveau de résolution sur le côté droit de la tête de mannequin. Le côté gauche de la tête a été reconstruit à un niveau de résolution très grossier, soit une résolution de 64 noeuds par côté. Notez à la hauteur de la bouche et du nez les fortes discontinuités visibles. Ce phénomène de discontinuités n'est pas visible sur le front et sur la zone située entre le nez et la bouche. Cela est dû au fait que la transition entre les niveaux de résolution a eu lieu dans des régions à faible courbure.

Cette simulation montre que pour minimiser les artéfacts de visualisation liés aux discontinuités, il est préférable de balayer l'objet de façon à ce que la transition entre les niveaux de résolution se fasse dans des régions ayant un faible niveau de détail. Il est à noter également que de très grandes discontinuités dans les régions frontières peuvent signifier un mauvais alignement des données. Nous reviendrons sur ces aspects de la visualisation au chapitre 5.

Conclusion

Ce chapitre a présenté la première étape nécessaire à la mise sur pied d'un système de modélisation interactive multirésolution. Cette première étape a consisté à mettre en place une structure de données multirésolution basée sur les champs vectoriels ainsi que les mécanismes nécessaires à sa gestion et à sa mise à jour. Le champ vectoriel multirésolution est basé sur un *octree*, correspondant à l'extension en trois dimensions d'un arbre binaire classique. L'idée maîtresse derrière le champ vectoriel multirésolution est de permettre la reconstruction de la surface à différents niveaux de résolution au lieu de représenter la surface seulement dans les noeuds feuilles de l'*octree* à un seul niveau de résolution. Cette modification permet donc à différents niveaux de résolution de cohabiter dans la même représentation. La taille de l'enveloppe de reconstruction de chaque niveau de résolution est fixée par une constante ε multipliée par la taille des *voxels* de chacun de ces niveaux. Ainsi, comme la taille des *voxels* est divisée en deux à chaque incrément de niveau, la taille de l'enveloppe de reconstruction est également divisée en deux. C'est cette différence de taille d'enveloppe entre les niveaux de résolution qui permet la reconstruction de la surface d'un objet à partir de différentes densités de points.

Nous avons également abordé les notions de cohérence et d'interprétation du champ vectoriel. L'interprétation d'un champ vectoriel consiste à lire les données qui y sont encodées pour en extraire de l'information sur la surface reconstruite. Un module de visualisation par lancer de rayons est une forme d'interprétation tout comme un module permettant l'extraction d'une triangulation de la surface encodée. Nous avons également vu qu'un champ vectoriel cohérent est un champ vectoriel qui pourra être interprété correctement, sans artéfacts dus à une mauvaise structure des données le composant. Pour pouvoir interpréter une surface cohérente, nous avons défini une *règle d'interprétation des données* ainsi qu'une *règle de continuité des niveaux de résolution*. D'une part, la règle d'interprétation déclare que pour tout point du volume, les algorithmes d'interprétation doivent utiliser le niveau de résolution valide le plus fin possible localement et ignorer les niveaux plus grossiers. D'autre part, la règle de continuité des niveaux

de résolution oblige la reconstruction de tous les niveaux de résolution intermédiaires entre le niveau le plus grossier et le niveau le plus fin valide et ce, pour tout point du volume contenant de l'information sur la surface. Ces deux règles assurent donc une désactivation progressive des niveaux grossiers pour laisser la priorité aux niveaux plus fins lorsqu'ils sont présents. La continuité des niveaux de résolution demande une surcharge au niveau du nombre de données brutes devant être recueillies. Cependant, il a été montré que cette surcharge ne dépasse pas 33% du nombre de données nécessaires à la reconstruction du niveau de résolution le plus fin.

Les expérimentations ont montré que la cohabitation des niveaux de résolution permet bel et bien la reconstruction d'une surface cohérente à partir de différentes densités de points. Cependant, l'expérience présentée dans ce chapitre posait l'hypothèse que l'utilisateur choisissait manuellement le niveau de résolution auquel devait être reconstruite la surface. Le prochain chapitre tente de pallier à cette limitation en proposant un algorithme de compression automatique temps réel pour permettre une sélection optimale du niveau de résolution durant l'acquisition des données.

Chapitre 4

Sélection automatique du niveau de résolution

4.1 Introduction

La plupart des systèmes de modélisation interactive proposés [4, 30, 25] reconstruisent le modèle 3D d'un objet à un seul niveau de résolution qui correspond généralement à la plus haute résolution permise par le système en fonction des ressources mémoire et processeur disponibles ainsi que de la résolution du capteur utilisé. Le choix du niveau de résolution peut également être fait en fonction du niveau de détail de la pièce à modéliser. Ce n'est qu'à la fin de l'acquisition des données brutes qu'il est possible de compresser le modèle pour en réduire la taille mémoire et faciliter son stockage ou son transfert. Avec ce type de système, le modèle doit être reconstruit à haute résolution en totalité avant d'activer le processus de compression. Cette approche de compression *hors ligne* n'est vraiment pas optimale, car le gain au niveau mémoire n'est obtenu qu'à la fin de l'acquisition et non pendant.

Une représentation volumétrique permettant la cohabitation des différents niveaux de résolution tout en conservant une représentation de surface *cohérente* a fait l'objet du chapitre précédent. Cette nouvelle représentation permettait de reconstruire une surface à partir de différentes densités de points. Cependant, les expériences montrées nécessitaient l'intervention de l'utilisateur pour choisir le niveau de résolution auquel les données étaient reconstruites.

Le défi actuel est donc de déterminer automatiquement, durant l'acquisition des

données brutes, le niveau de résolution optimal auquel doit être reconstruite la surface. Pour ce problème, le critère d'optimalité repose sur la *consommation de mémoire*. La surface doit être reconstruite le plus fidèlement possible en économisant au maximum les ressources mémoire, toujours dans le but de permettre une utilisation judicieuse de la mémoire. Comme nous le verrons dans la prochaine section, deux informations sont utilisées pour déterminer le niveau de reconstruction optimal de la surface. Ces deux informations sont la *densité locale de données* fournie par l'utilisateur ainsi que le *niveau de détail réel de la géométrie de l'objet.*

Dans un premier temps, la densité de points fournie par l'utilisateur permettra de contrôler la progression automatique des niveaux de résolution de grossier (N) vers fin ($N + P = N_{max}$). Pour les régions ayant atteint le niveau de résolution le plus fin, la condition basée sur la densité de points fera place à une condition basée sur le niveau de détail de la géométrie de l'objet qui permettra une compression des régions à faible courbure du niveau fin (N_{max}) vers le niveau grossier (N). Notez que les limites N et N_{max} correspondant respectivement aux niveaux de résolution le plus grossier et le plus fin pouvant être reconstruits sont fixées par l'utilisateur au début de la séance de modélisation. Le système présenté dans ce chapitre a fait l'objet d'un article de conférence [8].

Le chapitre abordera tout d'abord les deux mécanismes permettant de choisir de façon optimale le niveau de résolution auquel la surface doit être reconstruite. Ensuite, l'algorithme de compression en temps réel sera expliqué. La dernière section présentera des résultats expérimentaux obtenus avec le système proposé.

4.2 Reconstruction progressive des niveaux de résolution grossiers vers fins

Dans le contexte d'un système de modélisation interactive basé sur un capteur 3D tenu en main, l'utilisateur est au centre de la boucle de modélisation. Durant l'acquisition, il voit le résultat de la reconstruction du modèle 3D en temps réel, à l'écran. À partir du rendu de la surface en cours de reconstruction et de l'objet qu'il a sous les yeux, il décide de sa stratégie d'acquisition.

La question en suspens est donc comment l'utilisateur fait-il pour communiquer ses intentions au système de modélisation interactive ? Cette communication entre l'utilisateur et le système ne peut se faire que par l'intermédiaire de la densité locale de points.

Un mouvement rapide du capteur recueillera une faible densité de points tandis qu'un mouvement lent et persistant au-dessus d'une zone d'intérêt produira une forte densité de points.

Le premier mécanisme requis permettant de déterminer le niveau de reconstruction optimal de la surface repose sur une reconstruction progressive des niveaux de résolution grossiers vers les niveaux plus fins en fonction de la stratégie d'acquisition de l'utilisateur. L'idée est de débuter la reconstruction de la surface par le niveau de résolution le plus grossier. La reconstruction du niveau suivant ne sera permise que lorsque la surface encodée au niveau grossier sera considérée comme étant *stable*. La *stabilité d'une surface* est une condition permettant de déterminer si suffisamment de points ont été mesurés dans le voisinage d'une section de surface pour bien la décrire. Il est important de ne pas confondre la *stabilité de la surface* avec la *validité de la surface* présentée au chapitre 2. La différence entre ces deux conditions sera expliquée en détails dans la prochaine section. Une fois la *stabilité* atteinte pour un niveau de résolution, la reconstruction du niveau suivant est permise et le processus recommence jusqu'à ce que le niveau N_{max} soit atteint.

L'avantage de cette approche au niveau de la consommation mémoire repose sur le fait que le système ne permet la reconstruction des niveaux fins que si l'utilisateur le demande, en mesurant beaucoup de données pour une région de l'objet. Dans le cas contraire, comme lors de l'acquisition rapide d'un contexte présentant de faibles courbures, le système limite le niveau de résolution maximal, ce qui a pour effet d'économiser de la mémoire. La reconstruction progressive des niveaux de résolution repose sur la notion de *stabilité de la surface* qui sera abordée dans la prochaine section.

4.2.1 Stabilité de la surface

Lorsque l'utilisateur débute l'acquisition d'un objet, le système n'est pas en mesure de déterminer si la région en cours de balayage nécessite un faible ou un fort niveau de résolution. Le système ne peut prendre une décision que sur la représentation en cours de reconstruction dans le champ vectoriel et initialement, le champ est vide. Le système commence donc par reconstruire uniquement le niveau de résolution le plus grossier.

Au fur et à mesure que l'utilisateur ajoute des points au modèle, la représentation de la surface se reconstruit graduellement, toujours au niveau grossier. Lors de l'ajout de chaque nouveau point 3D, le système doit interpréter localement la représentation de la surface contenue dans le champ vectoriel et déterminer s'il permet ou pas la reconstruction du prochain niveau de résolution. La reconstruction du prochain niveau

de résolution ne sera permise que si le niveau grossier est localement *stable*. Voici maintenant une définition de la stabilité d'un noeud :

Stabilité d'un noeud : Un noeud *stable* est un noeud dont la densité d'échantillonnage est suffisante pour prévenir tout changement de la section de surface encodée lors de l'ajout de nouveaux points 3D.

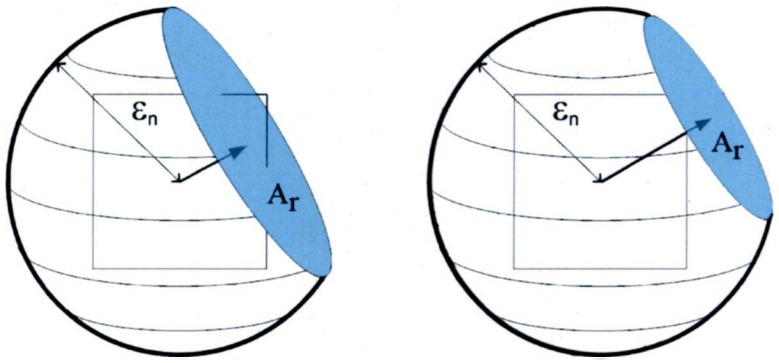
Une section de surface encodée dans un noeud stable peut être vue comme une section de surface qui ne devrait pas changer d'orientation ni de position lors de l'ajout de nouveaux points. Étant donné que l'état d'un noeud stable ne devrait pas changer, le système peut utiliser la représentation de la surface encodée pour prendre une décision. Dans le cas de la reconstruction progressive des niveaux de résolution, la décision que le système prend est simplement l'activation du niveau suivant. Cependant, nous verrons à la section 4.3 que le système peut utiliser la représentation d'une section de surface d'un noeud stable pour déterminer si la compression est possible dans cette région.

La condition permettant de déterminer si un noeud est *stable* est basée sur la densité de points 3D recouvrant sa section de surface. Si suffisamment de points ont participé à la reconstruction d'un noeud en fonction de la taille du plan tangent encodé, la valeur du champ vectoriel à cette position est considérée comme étant *stable*.

Il ne faut pas confondre *stabilité* et *validité* d'un noeud. La *validité* fait intervenir un test fait sur les valeurs propres de la matrice de covariance (section 2.4.3). La *stabilité* respose plutôt sur l'estimation de la densité des points 3D mesurés dans l'enveloppe de reconstruction. Dans les régions contenant de fortes courbures par rapport à la taille des *voxels* du niveau courant, un noeud ayant une densité d'échantillonnage suffisante peut être *stable* tout en n'étant pas *valide*. Cette distinction sera plus facile à comprendre après avoir abordé le principe d'estimation de la densité.

Estimation de la densité

Idéalement, l'estimation de la densité pour chaque noeud du champ vectoriel devrait tenir compte de la répartition des points bruts par rapport au plan tangent encodé. Malheureusement, comme le système ne conserve pas les données brutes, il n'est pas possible d'obtenir la répartition exacte des points, mais seulement une approximation via la matrice de covariance. D'un autre côté, le calcul de la densité pourrait se faire par un simple comptage du nombre de points 3D divisé par un facteur proportionnel à la taille de l'enveloppe de reconstruction du noeud en question. Cela revient à fixer un



(a) (b)

FIG. 4.1 – Exemples de l'intersection du plan tangent d'un noeud du niveau n avec son enveloppe de reconstruction de rayon ε_n. L'intersection A_r du plan tangent avec l'enveloppe de reconstruction est affichée en bleu pour une surface a) passant près du centre du noeud et pour une surface b) passant loin du centre du noeud.

seuil S_n pour chaque niveau de résolution. Lorsque le nombre de points participant à la reconstruction de la section de surface d'un noeud dépasse cette valeur, la représentation du champ vectoriel pour ce noeud est considérée *stable*.

Cette dernière approche est tout à fait valable, mais peut retarder la stabilisation de certains noeuds si leur distance par rapport à la surface de l'objet est élevée. Effectivement, comme le montrent les figures 4.1 a) et b), plus la surface est située loin du centre d'un noeud, plus l'aire de l'intersection de la surface de l'objet et de l'enveloppe de reconstruction du noeud diminue. Cela signifie que la probabilité qu'un point 3D soit mesuré à l'intérieur de l'enveloppe de reconstruction de ce noeud diminue également. Il est donc avantageux d'adapter le seuil S_n en fonction d'une estimation de la taille de la section de surface incluse à l'intérieur de l'enveloppe de reconstruction d'un noeud.

Pour estimer la taille de la section de surface de l'objet passant à l'intérieur de l'enveloppe de reconstruction d'un noeud, l'intersection entre le plan tangent encodé à ce noeud et son enveloppe de reconstruction est utilisée. La surface résultante de l'intersection, nommée A_r, est présentée en bleu à la figure 4.1 pour deux positions différentes. Modifier le seuil S_n en fonction de l'aire de A_r permet de favoriser une vitesse de stabilisation similaire peu importe la position du noeud par rapport à la surface de l'objet. Évidemment, dans le cas extrême où l'aire de l'intersection est minime, un point de grille ne peut devenir stable avant d'avoir recueillie un nombre minimum de données. Pour éviter cette situation, le seuil S_n est calculé de la manière suivante :

$$S_n = \frac{A_r}{A_{max}} K + K \tag{4.1}$$

où K est une valeur obtenue de manière empirique et A_{max} correspond à l'aire maximale que peut prendre l'intersection de l'enveloppe de reconstruction et du plan tangent lorsque ce dernier passe par le centre du noeud.

Stabilité VS Validité

Il est important de noter que le calcul de la densité de points à partir de l'aire du plan tangent compris dans l'enveloppe de reconstruction permet seulement d'accélérer la stabilisation des *noeuds valides*, c'est-à-dire, des noeuds ayant une représentation d'un plan tangent valide. Sans cela, il ne serait pas possible de calculer l'intersection entre l'enveloppe de reconstruction et la surface encodée, car cette dernière serait inexistante.

Si le noeud n'est pas valide, mais qu'il a été suffisamment échantillonné, il doit pouvoir être considéré comme étant *stable*. Sans cela, le processus de reconstruction progressive pourrait bloquer au niveau grossier dans les régions contenant de trop fortes courbures. Dans ces situations, il arrive parfois que le *test de validité* fait sur les valeurs propres de la matrice de covariance, présenté à la section 2.4.3, échoue constamment. Lorsque cette situation se présente, la densité de points requise pour atteindre la stabilité est calculée en utilisant $A_r = A_{max}$.

Cette explication montre que la notion de *stabilité* correspond à un niveau de confiance face à la qualité de l'échantillonnage d'une section de surface. Si la section de surface a été échantillonnée correctement, il est peu probable que son état évolue dans le futur, même après l'ajout d'un grand nombre de données. La notion de *validité* fait plutôt référence à la capacité de la représentation à encoder fidèlement la distribution des points mesurés dans l'enveloppe de reconstruction. Si la distribution des points à l'intérieur d'une enveloppe de reconstruction ne peut pas être approximée correctement par un plan tangent, le noeud ne deviendra jamais valide, mais il deviendra stable lorsque suffisamment de points y auront été mesurés.

4.2.2 Algorithme de reconstruction progressive

Pour reconstruire progressivement les niveaux de résolution, les étapes suivantes sont exécutées lors de l'ajout d'un nouveau point à la représentation :

1. Au niveau le plus grossier N, mettre à jour la représentation de la surface des noeuds appartenant à la cellule fondamentale du nouveau point ajouté.
2. Si un noeud mis à jour au niveau N est *stable*, reconstruire ses enfants si leur position est à l'intérieur de la cellule fondamentale du niveau N+1 du nouveau point ajouté.
3. Répéter récursivement l'étape 2 tant que les noeuds enfants appartiennent à des *niveaux de résolution actifs.*

Voici la définition d'un *niveau de résolution actif.*

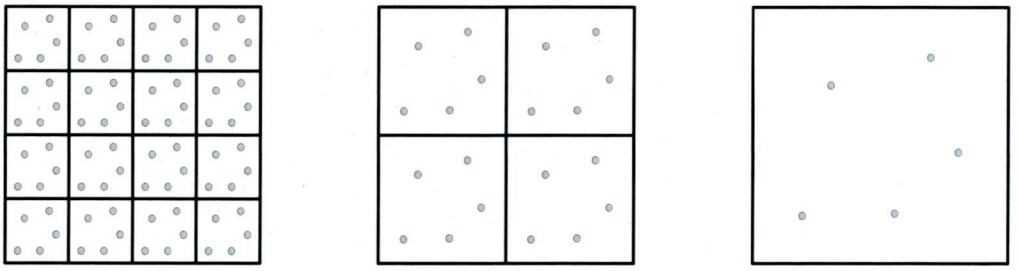
Niveau de résolution actif : Pour un noeud, un *niveau de résolution actif* est un niveau de résolution dont tous les noeuds parents des niveaux qui le précèdent dans la structure de l'*octree* sont stables, à l'exception du niveau N qui lui est toujours actif.

Les *niveaux actifs* sont donc l'ensemble des niveaux mis à jour lors de la reconstruction du champ vectoriel, à partir du niveau N et ce, jusqu'à ce qu'un niveau rencontré ne soit pas *stable.* Cela signifie que lors de l'ajout d'un nouveau point à la représentation, il est initialement ajouté au niveau grossier N. Ensuite, cette même donnée est ajoutée à chacun des enfants du niveau N si ce dernier est stable et si les noeuds enfants font partie de la cellule fondamentale, à leur niveau, du nouveau point ajouté. Le processus se répète tant que les enfants appartiennent à un niveau de résolution actif. Cela signifie que le dernier noeud auquel le nouveau point est ajouté n'a pas encore atteint la stabilité. Évidemment, la niveau N_{max} est le dernier niveau pouvant être reconstruit même s'il devient stable.

4.2.3 Surcharge due à la reconstruction progressive

Le mécanisme de reconstruction progressive des niveaux de résolution permet de sauver du temps et des ressources mémoire lors de la numérisation d'objets contenant des zones à faible courbure. Ces régions contenant peu de détail peuvent être balayées rapidement et la surface reconstruite uniquement aux niveaux de résolution grossiers.

Cependant, attendre que le niveau de résolution grossier soit complètement devenu stable avant de permettre la reconstruction du niveau suivant demande l'acquisition d'un plus grand nombre de points 3D. Ce phénomène est d'autant plus marqué dans les zones devant être reconstruites au niveau de résolution maximum, c'est-à-dire, au



(a) (b) (c)

FIG. 4.2 – Schéma montrant la progression du nombre de points par unité de surface en fonction des niveaux de résolution. a) Au niveau le plus fin, $16R$ points sont nécessaires pour stabiliser la surface. b) Au niveau précédent, $4R$ points suffisent pour stabiliser la surface. c) Au niveau le plus grossier, seulement R points sont requis pour obtenir une surface stable.

niveau N_{max}. Tous les points qui ont servi à stabiliser les niveaux N à $N_{max} - 1$ ne participeront pas à la reconstruction de la surface au niveau N_{max}, car le système ne conserve pas les données brutes. Les noeuds nouvellement activés sont initialement vides. Cela oblige l'utilisateur à rebalayer certaines zones de l'objet pour permettre la reconstruction des niveaux subséquents. Cependant, le surplus de points devant être recueilli par rapport à une approche reconstruisant l'objet au niveau N_{max} sans utiliser le mécanisme de reconstruction progressive n'est que de 33%.

Pour bien comprendre la surcharge imposée par le mécanisme de reconstruction progressive, prenons l'exemple d'une surface plane et posons que la stabilité est atteinte lorsqu'un certain nombre de points $S_N = R$ est capté dans l'enveloppe de reconstruction d'un noeud, peu importe le niveau de résolution. Pour cette démonstration, posons l'enveloppe de reconstruction à sa valeur minimale correspondant au *voxel* lui-même. Comme le montre la figure 4.2 a), pour une surface de taille donnée, le niveau le plus fin nécessite $16R$ points pour se stabiliser. Le niveau précédent montré à la figure 4.2 b) ne nécessite que $4R$ points pour atteindre la stabilité et le niveau le plus grossier n'a besoin que de R points pour se stabiliser. Somme toute, l'utilisateur a dû faire l'acquisition de $16R + 4R + R = 21R$ points pour reconstruire cette surface au niveau de résolution le plus fin en utilisant le mécanisme de reconstruction progressive. Sans le mécanisme de reconstruction progressive, la reconstruction de la surface au niveau le plus fin aurait nécessité l'acquisition de $16R$ données brutes. Pour ce scénario, la surcharge d'acquisition du mécanisme de reconstruction progressive est :

$$\frac{4R+R}{16R} = \frac{5}{16} = 0.3125 = 31.25\% \tag{4.2}$$

Pour généraliser ce résultat, posons que le nombre total de points nécessaire à la reconstruction du niveau le plus fin pour une surface donnée est K. Le niveau précédent nécessite $K/4$ points, le niveau d'avant $K/16$ et ainsi de suite jusqu'à ce que le niveau le plus grossier soit atteint. En développant cette série, nous obtenons dans le pire cas un surplus de :

$$\frac{\frac{K}{4}+\frac{K}{16}+\frac{K}{64}+\ldots}{K} = \lim_{N\to\infty} \sum_{i=1}^{i=N} \frac{1}{4^i} = \frac{1}{3} = 33.\bar{3}\% \tag{4.3}$$

Cela signifie que dans le pire des cas, le mécanisme de reconstruction progressive nécessite seulement l'acquisition de $33.\bar{3}\%$ plus de points, mais permet une gestion beaucoup plus efficace des ressources mémoire. De plus, dans un contexte de modélisation interactive, il est facile pour l'utilisateur de recueillir des données supplémentaires.

Il ne faut pas oublier que ce surplus n'est nécessaire que dans les zones devant être reconstruites au niveau de résolution le plus fin. Néanmoins, il est possible de limiter la perte des données brutes en permettant aux niveaux enfants d'accumuler des points avant que le parent n'ait atteint la stabilité. Par exemple, un niveau enfant pourrait accumuler des points lorsque son parent a atteint $\frac{S_N}{2}$ points. De cette manière, lorsque le parent a atteint la stabilité, l'enfant a pu profiter d'un certain nombre de points pour commencer à mettre à jour sa représentation de la surface. Cependant, même si les enfants ont la possibilité d'accumuler des données avant que leur parent ne soit stable, il est strictement interdit qu'ils deviennent stables avant leur parent. Ceci est une condition essentielle pour assurer la continuité des niveaux de résolution assurant une représentation multirésolution cohérente de la surface.

4.2.4 Facteurs influençant la densité de points pour atteindre la stabilité

La densité de points requise pour permettre la stabilité d'un noeud et ainsi assurer une reconstruction cohérente de la surface est fixée de manière empirique et peut dépendre du niveau de résolution. La densité de points dépend généralement du niveau de bruit des données fournies par le capteur, de la vitesse d'acquisition et du type de capteur.

L'influence de la vitesse d'acquisition et du type de capteur sur la densité de points requise est plus subtile que l'influence du niveau de bruit des données brutes. Prenons l'exemple d'un capteur 3D idéal recueillant des points 3D répartis uniformément sur la totalité de la surface de l'objet. En posant que chaque noeud nécessite un même nombre de points S_N pour se stabiliser, peu importe son niveau de résolution, la densité requise pour stabiliser la surface est quadruplée entre chaque niveau de résolution. Conserver un nombre de points constant entre les niveaux de résolution s'est avéré être une approche efficace pour ce type de capteur.

Lorsque l'échantillonnage des données brutes sur la surface de l'objet n'est pas uniforme selon toutes les directions, les seuils S_N doivent être déterminés avec plus de soin. Un exemple d'une telle situation est l'utilisation d'un capteur 3D tenu en main projetant un patron ayant la forme d'une trace laser. Avec ce type de capteur, la densité de points le long de la trace laser est beaucoup plus élevée que la densité de points selon l'axe de déplacement du capteur, généralement perpendiculaire à la trace laser. La densité selon l'axe de déplacement va dépendre de la vitesse du mouvement par rapport à la surface de l'objet. Pour ce type de capteur, la stratégie proposée pour fixer les seuils de stabilité S_N pour chaque niveau est de poser l'hypothèse que l'utilisateur, pour faire l'acquisition des zones de détail au niveau le plus fin, va balayer l'objet très lentement avec une densité de profils comparable à la densité de points sur la trace laser. Cela revient à fixer une densité d'échantillonnage uniforme pour le niveau de résolution le plus fin. En partant du niveau le plus fin, le seuil doit doubler à chaque niveau jusqu'au niveau le plus grossier pour pallier au fait que la densité selon la trace laser ne diminue pas lorsque le mouvement du capteur accélère. Sans cet ajustement des seuils, les noeuds du niveau grossier se stabiliseraient trop rapidement, après avoir été influencés par seulement quelques traces lasers. Cela signifie que des sections de surfaces seraient considérées stables et valides même si elles n'étaient pas échantillonnées adéquatement.

Pour terminer cette section, notez que des valeurs de seuil S_N très élevées ralentissent la progression, mais assurent une reconstruction plus robuste en présence de bruit. C'est à l'utilisateur de fixer ces seuils en fonction des conditions d'acquisition ainsi que du temps qui lui est alloué pour numériser l'objet. C'est un compromis entre rapidité d'acquisition et qualité de la reconstruction. Généralement, la densité est fixée aux environs de 40 à 100 points pour une enveloppe de reconstruction de rayon $\varepsilon = 2$.

Cette section a présenté le principe de reconstruction progressive des niveaux de résolution. Cette approche permet une économie de temps et de ressources mémoire lorsque l'objet modélisé comprend des zones de faible courbure. Malheureusement, cette technique à elle seule n'évite pas la consommation superflue de mémoire dans le cas où l'utilisateur recueillerait une forte densité de points dans une zone à faible courbure. Ce

problème de la compression temps réel des régions à faible courbure sera abordé dans la section suivante.

4.3 Compression temps réel des niveaux fins vers grossiers

Le mécanisme de reconstruction progressive des niveaux de résolution montre qu'il est possible d'économiser mémoire et temps en reconstruisant progressivement la surface de l'objet en fonction de la densité de points fournie par l'utilisateur. Cependant, ce mécanisme seul ne permet pas l'optimisation complète des ressources mémoire, car il détermine le niveau de reconstruction de la surface en se basant uniquement sur la densité de points fournie au système, sans tenir compte du niveau de détail réel de la géométrie de l'objet. Cela signifie qu'une zone à faible courbure balayée lentement sera reconstruite à un niveau de résolution élevé malgré le fait que le niveau grossier aurait été suffisant pour bien capter le peu de détail de la surface.

Cette situation est plus fréquente qu'il n'y paraît, car il est très difficile pour l'utilisateur de balayer lentement les régions d'intérêt sans déborder sur les surfaces planes les entourant. Prenons l'exemple d'un capteur 3D tenu en main projetant un patron laser de grande taille. En faisant l'acquisition d'un détail d'intérêt, il est certain que les zones à faible courbure entourant ce détail seront balayées lentement et reconstruites à un niveau de résolution trop élevé. Pour régler ce problème, un second mécanisme se basant sur le niveau de détail réel de la géométrie de l'objet doit participer au processus de sélection du niveau de résolution optimal de reconstruction de la surface. Ce second mécanisme est appelé *compression en temps réel* de la surface et permet une progression des niveaux fins vers les niveaux grossiers.

La figure 4.3 montre une section de surface d'un objet encodée par un noeud du niveau n et un ensemble de noeuds du niveau $n+1$. Dans cette figure, la surface de l'objet correspond à la courbe noire, les noeuds du niveau grossier n sont identifiés par un trait noir et les noeuds du niveau fin $n+1$, deux fois plus petits que ceux du niveau n, sont identifiés par des carrés en différents tons de gris. Si les deux représentations, grossière et fine, décrivent le même plan tangent à l'intérieur d'une certaine tolérance, c'est-à-dire qu'elles respectent la *condition d'ajustement*, ces données en double peuvent être compressées. Après la compression, la mémoire utilisée par les noeuds du niveau fin sera libérée et le noeud du niveau grossier sera considéré comme un noeud compressé. C'est en grande partie grâce à la mémoire libérée par le processus de compression en

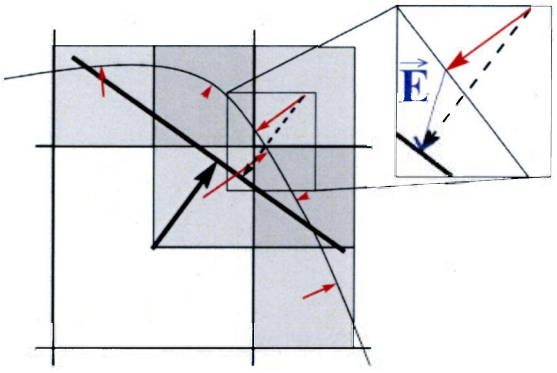


FIG. 4.3 – Surface encodée à deux niveaux de résolution différents. Le niveau n est représenté par le vecteur tracé en noir. Le niveau $n+1$ est représenté par les vecteurs tracés en rouge. Le vecteur $\vec{E}$ correspond à la différence entre les représentations de ces deux niveaux de résolution.

temps réel qu'il est possible avec une même quantité de mémoire de faire l'acquisition d'objets de plus grande taille ou à plus haute résolution.

4.3.1 Condition d'ajustement

Le mécanisme de compression temps réel utilise une *condition d'ajustement* pour la comparaison de deux représentations encodant la même surface à deux niveaux de résolution adjacents. Cette condition d'ajustement se base sur deux principes. Tout d'abord, il y a l'erreur commise entre les représentations de la même surface à des niveaux de résolution différents. Le second principe concerne les noeuds sur lesquels l'erreur doit être évaluée.

Erreur entre deux représentations adjacentes

L'erreur $\vec{E}$ commise entre la représentation de la même section de surface encodée par un noeud du niveau $n+1$ et un noeud du niveau n correspond à :

$$\vec{E} = F_{n+1}(v) - \tilde{F}_n(v) \tag{4.4}$$

- Le vecteur $F_{n+1}(v)$ représente le vecteur entre la position du noeud du niveau le plus fin $n+1$ et le point le plus près de la surface à ce niveau de résolution.

- Le vecteur $\widetilde{F}_n(v)$ représente le vecteur partant du centre du noeud du niveau $n+1$, mais rejoignant le point le plus près sur la surface encodée au niveau grossier n. Il ne faut pas confondre $\widetilde{F}_n(v)$ et $F_n(v)$. Le dernier terme correspond au vecteur partant du centre du noeud du niveau n et non du centre du noeud du niveau $n+1$.

Comme le montre la figure 4.3, la norme de la différence entre ces deux vecteurs encode l'erreur entre les deux représentations. C'est à partir de ce vecteur $\overrightarrow{E}$ que le mécanisme de compression en temps réel détermine si la compression doit avoir lieu ou pas.

Étant donné que la section de surface encodée dans les noeuds du niveau $n+1$ est plus petite que la section de surface encodée dans les noeuds du niveau n, avant de compresser une région de la surface, il faut considérer tous les noeuds du niveau fin étant à proximité du plan tangent encodé par la représentation grossière. L'identification des noeuds du niveau $n+1$ étant dans le voisinage d'un plan tangent encodé à un niveau grossier n repose sur le concept de *région de recouvrement.*

Région de recouvrement de la surface

La *région de recouvrement* d'un noeud au niveau n correspond à la région de l'espace contenant tous les noeuds du niveau $n+1$ qui ont la possibilité de représenter la même section de surface que celle encodée au niveau n. Formellement, la matrice de covariance $C(v)$ contenue dans un noeud du niveau n approxime localement la surface de l'objet par une section A d'un plan tangent de forme circulaire et de rayon égal à la demi-diagonale d'un *voxel.* Il ne faut pas confondre A avec l'intersection du plan tangent et de l'enveloppe de reconstruction du noeud qui est utilisée pour le calcul de la densité (section 4.2.1). La section A prend la même taille pour tous les noeuds d'un niveau n peu importe leur distance à la surface.

De manière plus spécifique, la *région de recouvrement* d'un noeud v au niveau n est définie de la manière suivante :

Région de recouvrement d'un noeud v au niveau n : Région du volume de travail défini par un cylindre dont la base correspond à la section de surface A encodée par le plan tangent, centrée sur $F_n(v)$ et dont la hauteur est de $2\varepsilon\Delta_{n+1}$.

La figure 4.4 montre un schéma de la *région de recouvrement* pour un noeud du

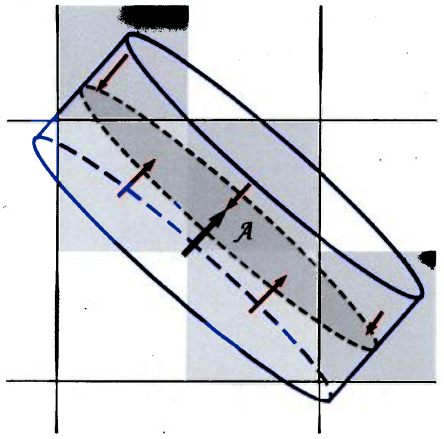


FIG. 4.4 – Région de recouvrement de la surface encodée au niveau grossier n. La région de recouvrement prend la forme d'un cylindre centré sur $F_n(v)$ et d'une hauteur de $2\varepsilon\Delta_{n+1}$.

niveau n. Tout d'abord, le vecteur tracé en noir correspond à $F_n(v)$. Notez que la *région de recouvrement* est centrée sur cette valeur. La base du cylindre qui définit la *région de recouvrement* correspond à la section de surface A encodée dans le noeud. La hauteur de la région de recouvrement est deux fois l'épaisseur de l'enveloppe de reconstruction du niveau suivant, c'est-à-dire, $2 * \varepsilon\Delta_{n+1}$. Pour comprendre pourquoi la hauteur de la *région de recouvrement* est fixée en fonction de la taille de l'enveloppe de reconstruction du niveau $n+1$, il faut poser l'hypothèse que l'erreur commise entre la représentation du niveau n et les données brutes est faible. Cette hypothèse est vraie dans les régions contenant de faibles courbures. Dans cette situation, les données recueillies sur la surface vont influencer les noeuds du niveau $n+1$ qui sont au plus à une distance de $\varepsilon\Delta_{n+1}$ de la surface.

Pour permettre la compression, tous les noeuds du niveau $n+1$ appartenant à cette région de recouvrement doivent représenter la même surface que celle encodée au niveau n à l'intérieur d'une certaine tolérance. Il est important de mentionner que les noeuds faisant partie de la région de recouvrement de la surface d'un noeud grossier ne sont pas nécessairement ses enfants dans la structure de l'*octree*. Généralement, lorsque la taille de l'enveloppe de reconstruction est de l'ordre de 2 ou 3 *voxels*, la région de recouvrement englobe des noeuds du niveau $n+1$ bien au-delà de la région de l'espace où se trouve le noeud grossier.

À partir de la définition de l'erreur $\overrightarrow{E}$ commise entre deux représentations et la définition de la *région de recouvrement* d'un noeud du niveau n, il est maintenant possible de définir la *condition d'ajustement* de la façon suivante :

Condition d'ajustement : La *condition d'ajustement* est respectée si 1) tous les noeuds du niveau $n+1$ appartenant à la région de recouvrement du noeud du niveau n encodent une représentation valide et stable et si 2) la norme de chaque erreur commise ($\|\overrightarrow{E}\|$) par tous ces noeuds du niveau $n+1$ est inférieure à un certain seuil E_{max} fixé par l'utilisateur.

La *condition d'ajustement* peut être divisée en deux partie. Tout d'abord, tous les noeuds du niveau $n+1$ appartenant à la *région de recouvrement* du noeud du niveau n doivent encoder une représentation *valide* et *stable* de la surface. La stabilité des noeuds du niveau $n+1$ est très importante dans un contexte de compression temps réel, car le système de modélisation interactive doit être en mesure de fonctionner avec un flot continu de données en provenance du capteur. La stabilité assure donc que les deux représentations de la surface ne révèleront pas de détails inattendus dans le futur. De plus, comme tous les noeuds du niveau $n+1$ doivent encoder une représentation *valide*, il est possible de calculer l'erreur commise $\overrightarrow{E}$ pour chacun d'eux. La nécessité de gérer un flot continu de données est la grande différence qu'il y a entre un mécanisme de compression en temps réel et un mécanisme de compression classique, appliqué sur le modèle haute résolution à la fin de l'acquisition.

La seconde partie de la *condition d'ajustement* concerne l'erreur maximale commise entre la représentation du niveau n et les représentations encodées dans les noeuds du niveau $n+1$ faisant partie de la *région de recouvrement*. Pour que la *condition d'ajustement* soit respectée, il faut que la norme de l'erreur commise par chacun des noeuds du niveau $n+1$ soit inférieure à un certain seuil E_{max}. Le seuil E_{max} est fixé empiriquement par l'utilisateur. Généralement, il correspond à 10% de la taille des *voxels*.

4.3.2 Algorithme de compression temps réel

Maintenant que la *condition d'ajustement* est bien définie, il faut déterminer quand il est possible de l'appliquer pour amorcer le processus de compression en temps réel. Déterminer le moment où le champ vectoriel peut amorcer sa compression est une étape critique. Démarrer la compression trop tôt pourrait générer un modèle final manquant certains détails de la surface et la démarrer trop tard entraînerait une utilisation inefficace des ressources mémoire. Pour être certain que le processus de compression soit activé au bon moment, les 4 conditions suivantes doivent être respectées avant d'amorcer le compression de la surface :

1. Le noeud grossier, au niveau n, doit avoir une représentation *valide* et *stable* de la surface.
2. Selon la *condition d'ajustement,* tous les noeuds du niveau $n+1$ appartenant à la *région de recouvrement* du noeud du niveau n doivent avoir une représentation *valide* et *stable* de la surface.
3. Tous les noeuds du niveau $n+1$ appartenant à la *région de recouvrement* du noeud grossier doivent être du *dernier niveau actif*, c'est-à-dire, soit du niveau le plus fin N_{max} ou être considérés comme des noeuds dont les enfants ont été retirés lors de la compression.
4. Selon la *condition d'ajustement,* l'erreur commise par chaque noeud du niveau $n+1$ appartenant à la région de recouvrement du noeud du niveau n doit être inférieure à E_{max}.

Condition 1 :

La première condition implique que le niveau grossier n, seule représentation restante suite à la compression, encode une représentation *valide* et *stable* de la surface. Cela signifie que le système considère que cette région de la surface a été suffisamment balayée pour permettre une reconstruction fidèle de la surface et le plan tangent encodé à ce noeud approxime bien le nuage de points qui a été mesuré dans son enveloppe de reconstruction. Il ne serait pas possible de compresser un *voxel* grossier instable ou invalide, car une fois la compression terminée, la représentation de la surface à cet endroit deviendrait incohérente.

Condition 2 :

La seconde condition assure que le plan tangent encodé au niveau grossier n est complètement recouvert par des noeuds du niveau $n+1$ qui eux aussi sont *stables* et *valides*. La validité de tous les noeuds du niveau $n+1$ appartenant à la région de recouvrement du niveau grossier signifie que l'erreur entre les représentations pourra être calculée partout dans le voisinage du plan tangent encodé au niveau n. La stabilité de tous ces noeuds du niveau $n+1$ assure également que l'ajout de nouvelles données à la représentation ne devrait pas modifier la surface encodée de manière significative. À cette étape, les noeuds du niveau $n+1$ faisant partie de la région de recouvrement et qui ont déjà été éliminés par le processus de compression sont considérés *valides* et *stables*.

Condition 3 :

La troisième condition a plusieurs implications. Tout d'abord, cette condition assure que suffisamment de points ont été recueillis sur l'objet pour reconstruire une représentation de la surface au niveau N_{max}. Pour bien comprendre cette première implication, prenons l'exemple de l'acquisition d'un objet correspondant à un sinus de faible amplitude. Au niveau N, c'est-à-dire au niveau le plus grossier permis pour la reconstruction, la surface reconstruite prendra l'allure d'un plan, car la taille des enveloppes de reconstruction à ce niveau est trop grande par rapport au détail de la surface. Au niveau $N+1$, le même phénomène se produit. Cela signifie que la représentation du niveau N et la représentation du niveau $N+1$ sont très similaires et respectent donc la *condition d'ajustement*. Dans une telle situation, faut-il compresser la représentation et ne garder que la surface encodée au niveau N ? La réponse à cette question est évidemment non, car il ne serait jamais possible de reconstruire le sinus de faible amplitude. Cet exemple met en évidence le fait que les décisions du mécanisme de compression sont prises sur la représentation de la surface contenue dans le champ vectoriel. Pour permettre la compression, il faut donc laisser la possibilité au système d'obtenir une représentation *stable* de la surface à la résolution maximale pour s'assurer que tous les détails de l'objet pouvant être obtenus par le capteur soient bien reconstruits. Une fois qu'une surface *stable* est reconstruite au niveau le plus fin, c'est-à-dire au niveau N_{max}, le système a la représentation de la surface la plus fidèle disponible dans les conditions d'acquisition imposées par l'utilisateur et il a donc la possibilité de prendre une décision face à la compression.

La seconde implication de cette troisième condition est que lorsque le processus de compression est démarré, des noeuds plus grossiers que le niveau N_{max} peuvent participer au processus seulement s'ils ont déjà été compressés. Ces noeuds sont considérés comme des noeuds feuilles de l'*octree* faisant partie du *dernier niveau actif*. Le respect de cette condition assure qu'aucun détail important ne sera oublié lors de la reconstruction de la surface.

Condition 4 :

La quatrième condition assure que le modèle compressé de la surface représente toujours la même surface que le modèle précédent à l'intérieur de la tolérance E_{max} de la *condition d'ajustement*.

Les quatre conditions précédentes assurent que le processus de compression ne

démarrera pas trop tôt et qu'aucun détail ne sera oublié sur la surface finale de l'objet. Pour éviter que le processus de compression ne démarre trop tard et ainsi consomme plus de mémoire qu'il n'en faut, ces quatre conditions doivent être testées à chaque fois qu'un nouveau point est ajouté au modèle. Une fois que les quatre conditions sont respectées, le noeud grossier du niveau n peut être compressé. Ses enfants dans la structure de l'*octree* ne sont plus nécessaires et peuvent être enlevés libérant ainsi de la mémoire qui pourra être utilisée pour la reconstruction d'un objet de plus grande traille. Le noeud grossier est identifié comme un noeud ayant été compressé et faisant dorénavant partie du dernier niveau actif. N'oublions pas qu'il contient déjà une représentation valide de la surface, car même s'il n'était pas du *dernier niveau actif*, il a été mis à jour à chaque fois qu'une nouvelle donnée a été ajoutée au modèle.

4.4 Compression hors ligne

La compression hors ligne est similaire à la compression en temps réel à l'exception qu'il est maintenant certain que l'utilisateur n'ajoutera pas de points 3D supplémentaires au modèle. L'utilisateur est satisfait de la représentation de la surface reconstruite qu'il voit à l'écran, fournie par le module de visualisation interactif. Cela signifie que la compression hors ligne n'a plus besoin de se préoccuper des noeuds enfants qui ne sont pas stables : ils n'auront jamais la chance de se stabiliser, car plus aucun point ne sera ajouté à la représentation. Pour cette même raison, il est également possible de compresser des régions de la surface n'ayant pas atteint le niveau N_{max}. Cela est possible car l'utilisateur a confirmé que le modèle contenait tous les détails de l'objet qu'il désirait. Donc, si deux niveaux adjacents encodent la même surface plane et respectent l'erreur maximale permise par la *condition d'ajustement*, il est avantageux de les compresser même si le plus fin d'entre eux n'a pas atteint le niveau N_{max}. La compression peut également avoir lieu même si la région de recouvrement n'est pas complète. Dans ce sens, la condition 2 est relaxée. Pour déterminer si une région peut être compressée, il suffit de tester les conditions 3 et 4 seulement sur les noeuds valides appartenant à la région de recouvrement du noeud grossier.

4.5 Expérimentations

4.5.1 Scénario d'acquisition typique

Nous avons validé les concepts présentés à partir de données de simulation et de données en provenance d'un capteur 3D réel. Le premier jeu de données étudié a été obtenu par simulation d'un capteur projetant une trace laser sur une triangulation haute résolution d'une tête de mannequin. Cet objet présente différents niveaux de détail. Les joues correspondent à des régions contenant de faibles courbures et le nez, les yeux et la bouche correspondent à des zones contenant beaucoup de détails. Pour chaque moment de l'acquisition, les points 3D ont été obtenus en calculant l'intersection entre le plan laser et les triangles du modèle du mannequin.

La simulation débute par l'acquisition du contexte et elle se poursuit par après avec la modélisation haute résolution des zones de détail, qui correspondent ici aux yeux, au nez et à la bouche du mannequin.

La figure 4.5 présente l'information fournie par le système durant cette séance d'acquisition. La ligne du haut montre un rendu en tons de gris de la surface pour différents temps de l'acquisition. L'image a été obtenue à l'aide du module de visualisation basé sur une technique de *lancer de rayons* qui sera présentée dans le prochain chapitre. La ligne du bas montre une vue alternative montrant le même modèle en fonction du niveau de résolution local de la surface. D'un côté, les teintes de rouge correspondent aux régions de la surface qui n'ont pas encore atteint le niveau de résolution N_{max} et qui sont dans la phase de reconstruction progressive des niveaux de résolution de grossiers vers fins. Le rouge foncé correspond au niveau de résolution le plus grossier et le jaune correspond au niveau de résolution le plus fin. D'un autre côté, les teintes de bleu apparaissent lorsque le mécanisme de compression en temps réel est activé. Les régions bleu foncées correspondent à des régions qui ont été compressées davantage. Au total, il y a quatre niveaux de résolution du plus grossier au plus fin.

Comme il est montré à la figure 4.5 a), la première étape consiste en l'acquisition du contexte. Le bas de la figure 4.5 a) montre qu'à cette étape, la reconstruction de la surface n'est permise qu'à un faible niveau de résolution à cause de la faible densité de points disponible. Le rendu en tons de gris montre effectivement que la surface a été reconstruite à un faible niveau de résolution, phénomène très visible dans les régions contenant du détail comme le contour du nez, des yeux et de la bouche. Notez également qu'il y a une petite région sur le haut de la tête du mannequin qui a réussi à atteindre le niveau de résolution $N+1$ (rouge clair). Cela est dû à la configuration du simulateur

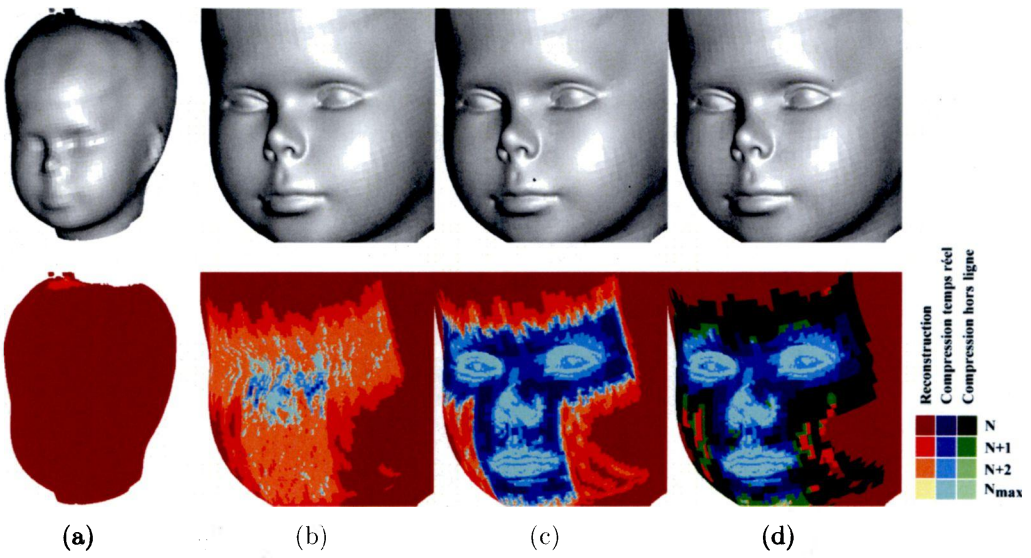


FIG. 4.5 – Représentation du modèle de la tête de mannequin affiché à l'utilisateur lors d'une séance d'acquisition typique. La ligne du haut montre le rendu de la surface en tons de gris. La ligne du bas affiche le même modèle en montrant le niveau de résolution local de la surface. a) Acquisition du contexte. b) Début de l'acquisition de la zone de détail. c) État du modèle immédiatement après la fin de l'acquisition. d) Modèle final obtenu après compression hors ligne. Les teintes de rouge sont utilisées pour montrer les zones dans la phase de reconstruction progressive des niveaux de résolution de grossiers vers fins. Les teintes de bleu sont utilisées pour les régions dans la phase de compression en temps réel. Les teintes de vert correspondent aux régions qui ont bénéficié de la compression hors ligne. Une teinte foncée correspond à un niveau de résolution grossier.

du capteur laser utilisé qui consiste à balayer l'objet en tournant autour de son centre. Ce mouvement circulaire crée un pôle aux extrémités du modèle fournissant ainsi une plus grande densité de points au module de reconstruction.

En fonction des rendus de la surface qui lui sont fournis par le système, l'utilisateur peut planifier sa séance d'acquisition dans le but d'obtenir un modèle de l'objet contenant des détails importants seulement aux endroits désirés. La seconde étape de la simulation consiste donc à augmenter la densité de points autour des régions d'intérêt. Comme il est présenté à la figure 4.5 b), les régions plus densément balayées autour des yeux, du nez et de la bouche apparaissent en teintes de rouge plus claires. Il y a même certaines régions qui ont recueilli une quantité suffisante de données pour stabiliser le niveau de résolution N_{max}. Ces régions apparaissent en bleu clair. Noter qu'à cette étape, le rendu en ton de gris montre un niveau de détail nettement supérieur dans les régions d'intérêt comparativement à l'étape précédente.

La figure 4.5 c) montre un modèle satisfaisant de la tête de mannequin à la fin de la séance d'acquisition. Les régions à faible courbure entourant les zones contenant du détail ont été compressées à un niveau de résolution grossier et sont présentées en teintes de bleu foncé. Comme il est visible dans l'image supérieure, malgré l'activation du mécanisme de compression en temps réel, les détails fins contenus dans les régions d'intérêt sont préservés et apparaissent en bleu clair. Un phénomène problématique précédemment mentionné est clairement visible à cette étape de la modélisation. Le pourtour des régions balayées lentement présente des zones sur lesquelles la densité de points est insuffisante pour permettre la stabilisation du niveau de résolution le plus fin. Cela signifie que localement, le processus de compression en temps réel n'a pas pu être enclenché. Ces régions sont donc reconstruites à un niveau de résolution potentiellement trop élevé pour le niveau de détail de la surface de l'objet. De plus, si l'utilisateur tente de les faire "disparaître" en rajoutant des données, il risque de propager le phénomène à d'autres régions.

La solution à ce problème réside en la compression hors ligne qui correspond à la dernière étape d'un scénario d'acquisition typique. Cette étape permet le nettoyage de la surface, car à ce moment, le système est certain qu'aucun nouveau point 3D ne sera ajouté au modèle. En terminant l'acquisition et en activant la compression hors ligne, l'utilisateur confirme qu'il est satisfait avec l'état actuel du modèle. La figure 4.5 d) montre le résultat final, après la compression hors ligne. Notez que les régions problématiques dans le pourtour de la zone d'intérêt ont été nettoyées correctement. Les régions n'ayant pas atteint le niveau N_{max}, mais satisfaisant la condition d'ajustement n'apportent pas d'information supplémentaire sur la surface et peuvent être compressées. Le résultat final montre bien que la surface de l'objet a été reconstruite

au niveau optimal en fonction de la densité de points fournie par l'utilisateur lors de l'acquisition du contexte et du niveau de détail réel de l'objet pour les régions contenant du détail.

4.5.2 Gestion de la mémoire

La simulation précédente a montré un scénario d'acquisition typique avec le système de modélisation interactive proposé et a aussi donné un aperçu qualitatif des gains mémoires pouvant être espérés lors de l'utilisation d'un tel système. La particularité de ce système est que le mécanisme de compression en temps réel est appliqué localement et progressivement pour libérer de la mémoire durant l'acquisition et ainsi permettre de mieux gérer la ressource. La prochaine expérience permet de chiffrer le gain au niveau de la consommation mémoire du système proposé. Cette expérience repose sur les données du modèle 3D du *Bunny* de Stanford et présente trois différentes approches d'acquisition. La première approche consiste à faire un échantillonnage uniforme du modèle sans le mécanisme de compression en temps réel. La seconde approche consiste également à échantillonner uniformément le modèle, mais cette fois-ci, en activant la compression en temps réel. Finalement, la dernière approche utilise toujours la compression en temps réel, mais de paire avec une stratégie d'échantillonnage différente : l'objet est échantillonné uniformément, mais partie par partie en recueillant suffisamment de points à chaque zone pour permettre la stabilisation du niveau de résolution le plus fin et ainsi permettre au mécanisme de compression en temps réel de montrer tout son potentiel. Notez que pour cet objet, l'étape de l'acquisition du contexte a été omise. Le *Bunny* est considéré comme un objet ayant un niveau de détail fin sur toute sa surface. La figure 4.6 montre la consommation de mémoire pour ces trois différentes approches en fonction du nombre de points ajoutés au modèle.

La courbe bleue de la figure 4.6 montre la progression de la consommation de mémoire, en nombre de noeuds utilisés dans l'*octree*, en fonction du nombre de points ajoutés au modèle pour l'approche d'acquisition utilisant un échantillonnage uniforme mais n'utilisant pas le mécanisme de compression en temps réel. Dans ce cas, des points 3D échantillonnés uniformément sur toute la surface de l'objet sont ajoutés au modèle progressivement. Comme la compression en temps réel n'est pas activée, la consommation de mémoire ne fait qu'augmenter jusqu'à ce qu'elle se stabilise. Comme le montre le graphique, cette stabilisation apparaît après l'ajout d'environ 2 millions de points au modèle. À partir de ce moment, la surface a débuté sa reconstruction partout au niveau N_{max}, car le nombre de noeuds actifs n'augmente presque plus jusqu'à la fin de la séance d'acquisition. Les données supplémentaires ajoutées au modèle servent à assurer la stabilité du niveau le plus fin. Pour la suite de l'expérience, la courbe bleue

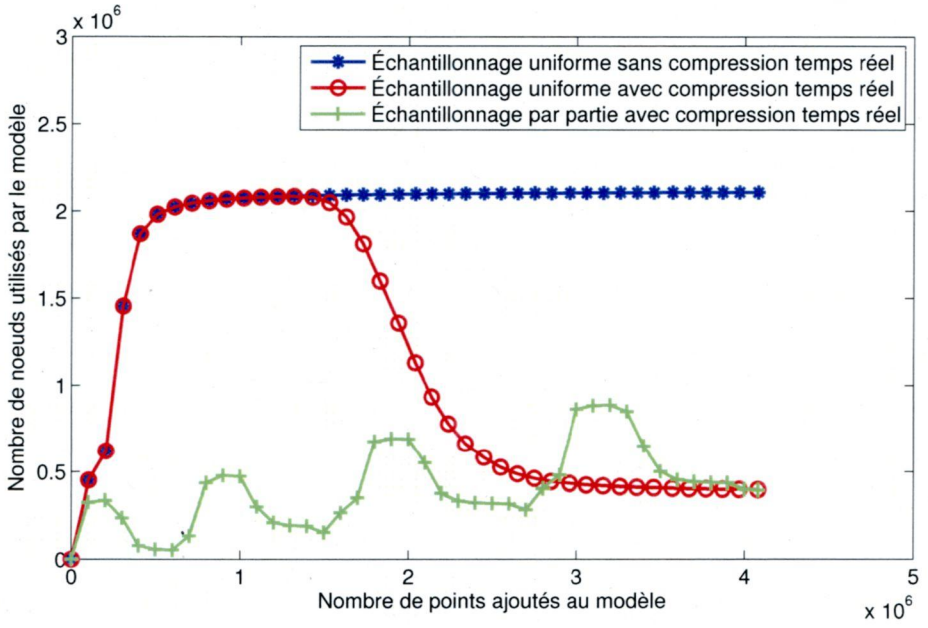


FIG. 4.6 – Consommation de mémoire en fonction du nombre de points 3D ajoutés au modèle lors de la reconstruction du *Bunny* pour trois différentes approches d'acquisition.

servira de référence, car elle représente le pire cas au niveau de la consommation de mémoire.

La stabilisation de la consommation de mémoire pour une surface donnée est une caractéristique d'une représentation volumétrique de type implicite comparée à une représentation de type explicite tel qu'un nuage de points. Dans le cas du nuage de points, la consommation de la mémoire ne fait qu'augmenter linéairement tout au long de l'acquisition. La représentation volumétrique a l'avantage de limiter la consommation de mémoire pour une surface donnée, car les points 3D ne sont pas conservés. Il faut plutôt voir le processus de reconstruction d'une surface à partir de points 3D comme une interprétation des données brutes pour en obtenir une représentation de plus haut niveau.

La courbe rouge de la figure 4.6 montre la même relation que la courbe bleue pour une stratégie d'acquisition uniforme avec le mécanisme de compression en temps réel activé. Au début de l'acquisition, la consommation de mémoire est similaire à celle observée pour un échantillonnage uniforme sans compression, car aucune région de la surface de l'objet n'a atteint le niveau N_{max} et le processus de compression en temps réel n'est pas encore activé. Remarquez qu'avec cette approche d'échantillonnage uni-

(a) (b) (c) (d)

FIG. 4.7 – Différents états du modèle du *Bunny* reconstruit en utilisant une stratégie d'échantillonnage uniforme (ligne du haut) et une stratégie d'échantillonnage par parties (ligne du bas). a) 0.5 million de points, b) 1.5 million de points, c) 2.5 millions de points, d) modèle final. Les tons de rouge et bleu ont la même signification que ceux utilisés dans la figure 4.5.

forme, la totalité de la surface progresse au même rythme. Cela signifie que le niveau N_{max} est stabilisé à peu près en même temps pour toutes les régions de l'objet. Ce moment correspond à l'instant où la consommation de mémoire commence à diminuer, c'est-à-dire après l'ajout de 1.8 million de points 3D dans ce cas particulier. Cette diminution de la consommation de mémoire est due au fait que la compression en temps réel peut s'amorcer et ainsi libérer de la mémoire. Malgré le gain considérable au niveau mémoire obtenu à la fin de l'acquisition, cette approche n'apporte pas vraiment d'avantage comparativement à une approche n'utilisant pas le mécanisme de compression en temps réel, car la consommation maximale de mémoire n'est à peu près pas réduite. Le système consomme toujours beaucoup de mémoire car il doit reconstruire en totalité la surface de l'objet à un niveau de résolution élevé avant que ne s'amorce la compression. Cette observation montre que le mécanisme de compression en temps réel nécessite une stratégie d'acquisition différente pour exploiter tout son potentiel.

La stratégie d'acquisition permettant d'exploiter tout le potentiel du mécanisme de compression en temps réel est celle utilisée pour la simulation avec la tête du mannequin. Il est possible d'obtenir une réduction très appréciable de la mémoire utilisée en ajoutant, partie par partie, des sections de l'objet au modèle. Dans ce scénario, il faut recueillir suffisamment de points à chaque partie pour permettre la stabilisation locale

	Fin de l'acquisition	Consommation maximale	Après la compression hors ligne
Sans compression temps réel	2.110M	2.110M	0.343M (83.7%)
Échantillonnage uniforme	0.396M (81.2%)	2.080M (1.1%)	0.317M (84.9%)
Échantillonnage par parties	0.394M (81.3%)	0.880M (57.9%)	0.316M (85.0%)

TAB. 4.1 – Consommation de mémoire pour les trois différentes approches de modélisation étudiées. La première colonne correspond au nombre final de noeuds utilisés pour chaque modèle. La seconde correspond au nombre maximum de noeuds utilisés durant l'acquisition et la dernière colonne correspond à l'état du modèle suite à la compression hors ligne. Les taux de compression, entre parenthèses, sont obtenus à partir de la consommation de mémoire du modèle fait sans l'activation du mécanisme de compression en temps réel.

du niveau de résolution le plus fin et ainsi permettre l'activation du mécanisme de compression en temps réel. La courbe verte de la figure 4.6 montre le gain au niveau de la consommation mémoire pour une telle stratégie appliquée sur le modèle du *Bunny*. Pour ce faire, le modèle a été séparé en quatre sections différentes et la reconstruction de la suivante n'a été amorcée qu'après l'ajout d'un grand nombre de points sur la partie courante pour permettre sa reconstruction à un niveau de résolution élevé. La figure 4.7 montre la progression du modèle du *Bunny* pour l'approche d'acquisition avec un échantillonnage uniforme (ligne du haut) et par parties (ligne du bas), toujours en utilisant le mécanisme de compression en temps réel. Notez que pour l'approche par parties, la surface de l'objet a pu amorcer sa compression plus tôt et ainsi libérer une grande quantité de mémoire.

La table 4.1 résume les avantages de l'utilisation du système d'acquisition proposé avec une stratégie d'acquisition par parties. La première colonne correspond à la consommation de mémoire finale en nombre total de noeuds utilisés par chaque modèle, à la fin de l'acquisition, mais avant l'activation de la compression hors ligne. Ce tableau montre que les deux stratégies d'acquisition présentent des taux de compression similaires qui sont de l'ordre de 80%. Ce qui distingue les deux stratégies d'acquisition, c'est la consommation maximale de mémoire tout au long du processus de reconstruction de la surface. Dans le cas de l'échantillonnage uniforme sur l'ensemble de l'objet, la consommation maximale de mémoire atteint 2.080M de noeuds, c'est-à-dire un gain de seulement 1.1% comparativement à une acquisition conventionnelle, sans compression en temps réel. D'un autre côté, l'échantillonnage par partie présente une consommation de mémoire beaucoup plus intéressante qu'un échantillonnage uniforme de l'objet, car la consommation maximale en tout moment de l'acquisition est passée de plus de 2M de noeuds à seulement 0.880M. Cette diminution correspond à un gain de 57.9% comparativement à une acquisition conventionnelle.

La troisième colonne du tableau montre les taux de compression après l'application de la compression hors ligne sur chacun des trois modèles. Comme l'objet a été

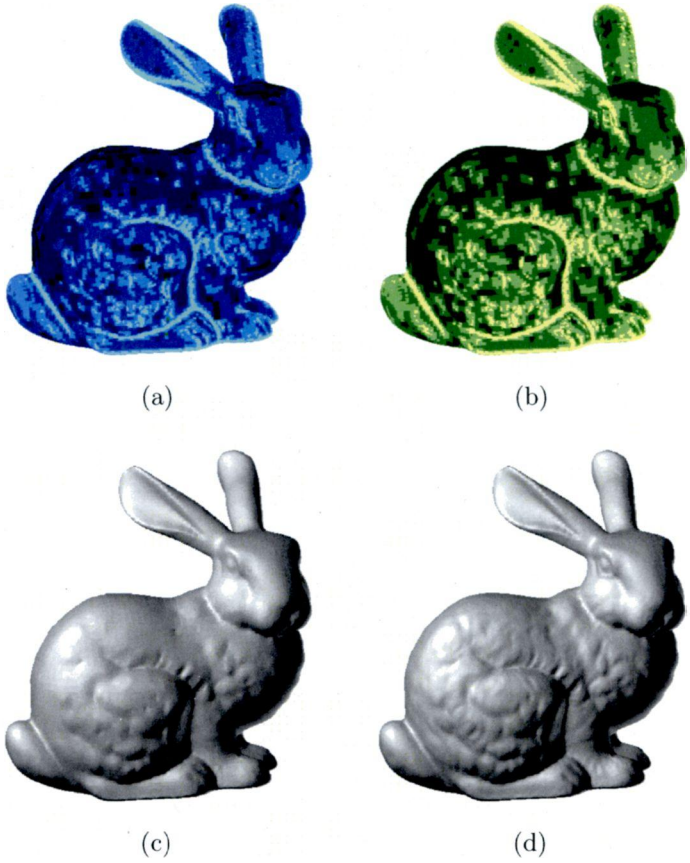


FIG. 4.8 – Niveaux de résolution local pour le modèle du *Bunny* après compression hors ligne du modèle obtenu par a) la stratégie d'échantillonnage uniforme avec compression en temps réel et b) la stratégie d'échantillonnage uniforme sans compression en temps réel. Les tons de vert et bleu ont la même signification que ceux utilisés à la figure 4.5. c) Rendu obtenu à partir du modèle présenté en a). d) Rendu du modèle de référence (sans compression en temps réel, ni hors ligne).

densément échantillonné sur toute sa surface, le gain qu'apporte la compression hors ligne comparativement à la compression en temps réel n'est que de 3.5%. Cela s'explique par le fait que pour cet objet, la grande majorité de la surface a atteint le niveau de résolution N_{max} permettant ainsi au mécanisme de compression en temps réel d'optimiser les ressources mémoire. Très peu de zones sont influencées par le mécanisme de compression hors ligne.

La figure 4.8 a) montre le modèle du *Bunny* pour une stratégie d'échantillonnage uniforme avec compression en temps réel suivie de la compression hors ligne. Notez certaines zones sur le dos du modèle maintenant affichées en vert foncé signifiant que ces régions ont été influencées par la compression hors ligne. La figure 4.8 b) montre le résultat de la compression hors ligne sur le modèle du *Bunny* obtenu par une stratégie

d'échantillonnage uniforme sans compression en temps réel. Une observation attentive du creux de l'oreille et du bout du nez du *Bunny* permet de voir que certains détails supplémentaires ont été conservés sur le modèle n'ayant subi que les effets de la compression hors ligne. Ces détails supplémentaires expliquent la différence d'environ 1.2% entre les taux de compression obtenus par la compression hors ligne du modèle n'ayant pas subi au préalable une compression en temps réel et les deux autre modèles.

Cette différence au niveau des taux de compression hors ligne s'explique par le fait que le mécanisme de compression en temps réel ne peut qu'estimer la stabilité du *voxel*. Lors de l'ajout de points 3D à la représentation, le système doit déterminer si les données contenues dans un noeud risquent de changer dans le futur. Cette question est résolue grâce à la condition de stabilité énoncée à la section 4.2.1. Cependant, il faut comprendre que cette condition ne se base que sur une approximation de la stabilité réelle des données et affirme que la surface *ne devrait pas* évoluer avec l'ajout de nouveaux points. Cette condition n'est malheureusement pas infaillible. Le fait d'appliquer la compression hors ligne sur un modèle n'ayant pas été compressé par un mécanisme de compression en temps réel peut être vu comme en renforcement de la condition de stabilité. Effectivement, la compression hors ligne est appliquée seulement après que tous les points 3D disponibles pour cette simulation aient été ajoutés au modèle. Étant donné que tous les points 3D disponibles ont participé à la reconstruction de la surface au niveau N_{max} avant d'amorcer quelque compression que ce soit, il se peut que certaines régions qui ont été considérées comme compressibles par le mécanisme de compression en temps réel ne soient pas compressibles par le mécanisme de compression hors ligne à cause de l'apparition d'un petit détail supplémentaire qui est venu seulement avec l'ajout de la totalité des points 3D. Ce sont ces petits détails supplémentaires captés par le modèle n'ayant subi que la compression hors ligne qui influencent les taux de compression.

La figure 4.8 c) montre le modèle du *Bunny* présenté en tons de gris pour une stratégie d'échantillonnage uniforme avec compression en temps réel suivie par une compression hors ligne. La figure 4.8 d) présente le rendu du modèle de référence du *Bunny*, sans aucune compression. Ces deux figures permettent d'apprécier l'effet de la compression sur le niveau de détail du modèle, effet très présent sur le dos du *Bunny* ainsi que dans les détails de son pelage.

4.5.3 Reconstruction à haute résolution

La dernière simulation de reconstruction de surface à partir de données synthétiques a pour but de montrer que le système présenté permet d'atteindre localement un niveau

de résolution plus élevé que les approches n'utilisant qu'un seul niveau de résolution. Cette simulation consiste à reconstruire la tête de mannequin en utilisant un niveau de résolution très élevé pour les détails fins du modèle qui, dans ce cas-ci, correspondent aux yeux. Au niveau le plus fin, la résolution de chaque noeud de la grille est d'environ 50 microns, soit 4096 noeuds pour une dimension de 200mm. Reconstruire certaines régions à un si haut niveau de résolution à l'aide d'un ordinateur de table n'est possible que grâce à la représentation multirésolution et à ses capacités de compression durant la cueillette des données.

Sans cette nouvelle approche de modélisation interactive, pour reconstruire les détails, il aurait fallu reconstruire la totalité du modèle à l'aide d'un *octree* d'une résolution de 4096 noeuds par côté. Cette résolution correspond au niveau 12. Même grâce à un *octree*, qui est très efficace au niveau mémoire pour encoder une surface dans un volume, cela aurait demandé une quantité de mémoire non disponible sur des ordinateurs de table courants. Pour estimer la quantité de mémoire requise pour encoder une surface dans un octree monorésolution du niveau 12, approximons la surface à encoder par une sphère occupant 90% du volume disponible. Cette sphère aurait un rayon de 1800 unités et sa surface une aire d'environ 40 millions d'unités carrées. En tenant compte d'une enveloppe de reconstruction de taille $\varepsilon = 2$, nous aurions besoin d'approximativement 160 millions de noeuds sans compter ceux nécessaires à gérer la structure de l'*octree*. Ces 160 millions de noeuds nécessiteraient plus de 10Go de mémoire. À cause de cette grande consommation de mémoire, les méthodes utilisant un niveau de résolution unique se limitent généralement à une résolution de 1024 noeuds par côté de l'*octree*. Cependant, l'utilisation d'un *octree* multirésolution règle ce problème en permettant à certaines régions d'être reconstruites à un niveau de résolution faible et à augmenter le niveau de résolution dans les régions contenant de fortes courbures.

Pour cette simulation, la stratégie d'acquisition utilisée diffère un peu de celle précédemment employée. La différence se situe au niveau de l'acquisition du contexte où le niveau N_{max} a été fixé à un niveau inférieur, soit le niveau 8. Cela permet au mécanisme de compression en temps réel de s'activer sans nécessairement être obligé de reconstruire le contexte à un très haut niveau de résolution. En plus d'économiser de la mémoire, cette approche nécessite une densité de données 3D beaucoup plus faible.

La première étape a consisté à balayer rapidement le contexte et à le reconstruire à un niveau de résolution grossier $N_{max} = 8$. Cela correspond à une résolution de 256 noeuds par côté de l'*octree*. Cette étape d'acquisition rapide du contexte a nécessité 3.7 millions de points 3D et le modèle obtenu nécessitait aux environs de 160000 noeuds, dont 100000 du niveau 7 et 42000 du niveau 8. Les autres noeuds étaient utilisés pour consolider la structure de l'*octree*.

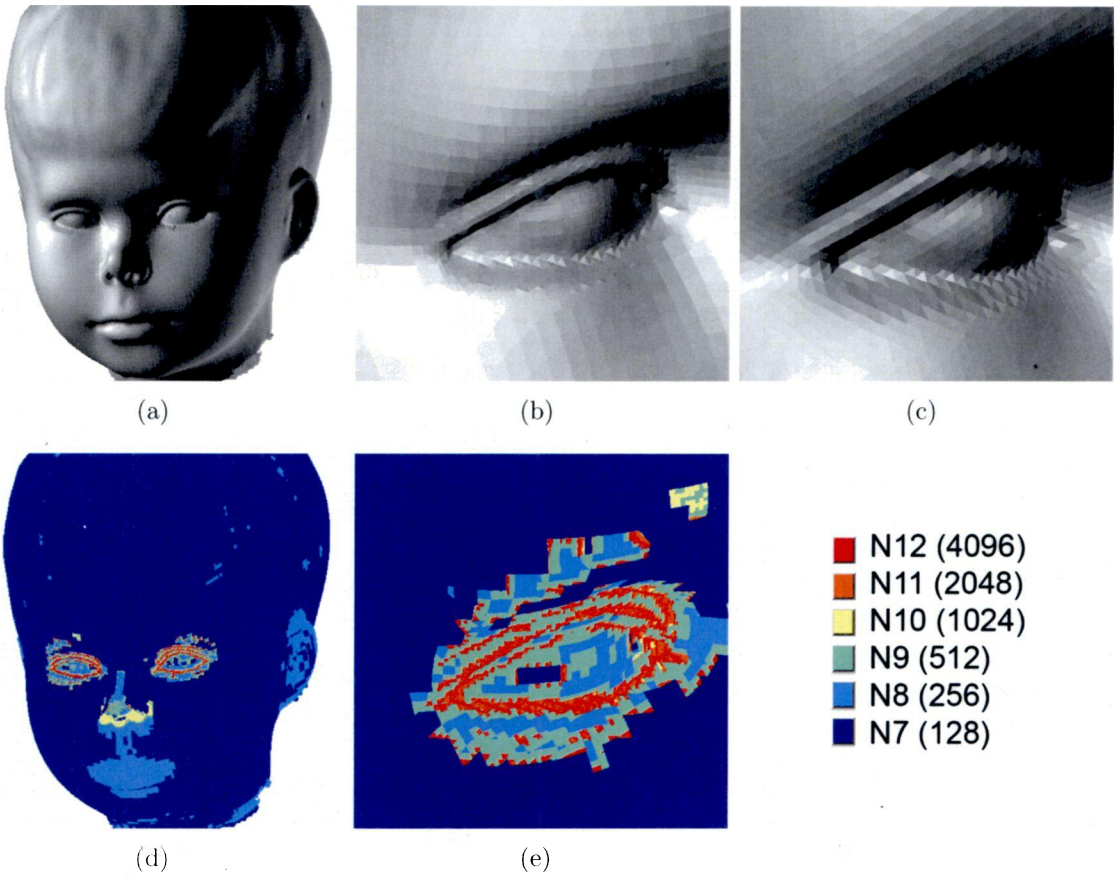


FIG. 4.9 – Résultats de la simulation de reconstruction de la tête de mannequin à un niveau de résolution élevé. a) État du modèle à la fin de la séance d'acquisition. b) Vue rapprochée de l'oeil droit de la tête de mannequin pour montrer le niveau de détail de la surface reconstruite. Notez qu'il est possible de voir les arêtes des triangles du modèle utilisé par le simulateur. c) Rendu fait à l'aide d'un moteur 3D de la triangulation de la tête de mannequin utilisée par le simulateur avec un point de vue similaire à celui utilisé en b). d) Niveau de résolution local pour le modèle 3D avec le même point de vue que celui utilisé en a). Il est important de noter que la signification des couleurs diffère de celle utilisée lors des expériences précédentes. Les régions bleu foncées correspondent à celles encodées au niveau 7 et les régions rouges correspondent à celles encodées au niveau 12. e) Niveau de résolution local pour la vue rapprochée présentée en b).

La seconde et dernière étape a été d'augmenter considérablement la densité de points dans la région entourant les yeux de la tête de mannequin. Pour cette étape, le niveau N_{max} a été fixé à 12, soit une résolution de 4096 noeuds par côté de l'*octree*. Cela signifie que chaque noeud est distancé de 50 microns. La figure 4.9 a) montre l'état du modèle suite à cette étape, en tons de gris. La figure 4.9 d) montre le même point de vue mais en affichant les niveaux de résolution en différentes couleurs. Ici, il est important de noter que les différentes couleurs n'ont pas la même signification que lors des dernières expérimentations. Les régions en bleu foncé correspondent aux régions encodées au niveau de résolution le plus grossier, soit le niveau 7. Les régions en rouge correspondent à celles encodées au niveau de résolution le plus fin, soit le niveau 12. La consommation finale de mémoire est de 625000 noeuds dont plus de 60% sont utilisés dans les deux derniers niveaux de résolution. La reconstruction des zones de détail a nécessité plus de 16 millions de données brutes supplémentaires.

Les figures 4.9 b) et e) montrent une vue rapprochée de l'oeil droit de la tête de mannequin dans le but d'apprécier le niveau de détail pouvant être atteint. Remarquez dans le pourtour de l'oeil que les arêtes des triangles adjacents ayant des normales différentes ont été encodées au niveau 12, soit le niveau le plus fin disponible. Les régions à faible courbure entourant ces zones de détails ont été compressées par le mécanisme de sélection automatique du niveau de résolution. La figure 4.9 c) présente un rendu fait à l'aide d'un moteur 3D de la triangulation de la tête de mannequin, utilisée par le simulateur, avec un point de vue similaire à celui utilisé pour générer la figure 4.9 b). Cette comparaison montre que la grande partie des détails contenus dans la triangulation ont été bien encodés dans la représentation volumétrique.

Une des difficultés de cette simulation a été de générer une densité de points suffisante pour permettre l'activation et la reconstruction du niveau le plus fin, soit le niveau 12. Il n'aurait pas été possible d'atteindre cette densité avec le capteur 3D tenu en mains présenté au chapitre 2. Ce capteur présente un niveau de bruit trop élevé qui est de l'ordre de grandeur de la distance entre les noeuds de l'*octree* au niveau le plus fin. Pour permettre la reconstruction de la surface à cette résolution, le niveau de bruit des données brutes doit être inférieur à la distance entre les noeuds pour que les données recueillies dans les enveloppes de reconstruction soient représentatives de la surface. Lorsque le niveau de bruit est trop élevé par rapport à la taille des enveloppes de reconstruction, les noeuds deviennent rarement valides et dans ces rares cas, il y a souvent présence d'un phénomène de surface double.

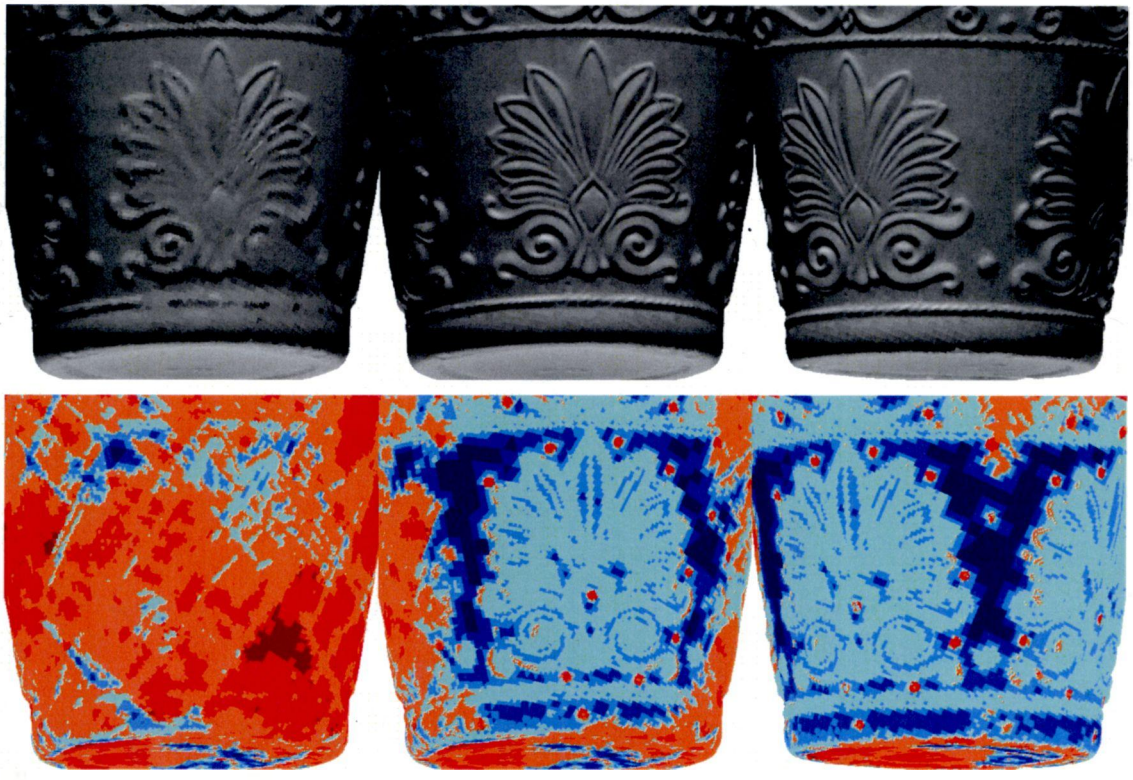
4.5.4 Reconstruction à partir de données réelles

Nous avons aussi testé le mécanisme de compression temps réel avec des données réelles. Ce jeu de données a été obtenu à l'aide d'un HandyScan de Creaform. Le HandyScan correspond à la version commerciale du capteur 3D développé par le groupe de Vision 3D du Laboratoire de Vision et Systèmes Numériques de l'Université Laval. Son fonctionnement a été abordé en détails au chapitre 2.

Pour cette nouvelle expérience, la séance d'acquisition a été divisée en deux, soit l'acquisition du contexte et l'acquisition des régions contenant du détail. La figure 4.10 a) montre l'état du modèle du vase après une capture rapide du contexte. Le vase a une dimension de 70 cm de haut par 50 cm de diamètre. La ligne du haut présente un rendu du modèle en tons de gris. La ligne du bas montre le niveau de résolution local du modèle en utilisant la même légende que pour la figure 4.5. Sauf en de rares endroits, la densité de points recueillie durant cette étape est insuffisante pour amorcer la compression temps réel. La figure 4.10 b) présente l'état du modèle suite à l'ajout d'un premier détail. La densité de points dans cette région est suffisante pour permettre la reconstruction du modèle au niveau de résolution le plus fin et aussi, activer la compression temps réel pour éviter de gaspiller la mémoire dans les régions à faible courbure. Notez que les régions contenant peu de détail, situées autour de la zone d'intérêt, ont été compressées avec succès et sont présentées en bleu foncé. La figure 4.10 c) montre l'état du modèle après l'ajout d'une seconde région d'intérêt. Ici aussi, les régions à faible courbure ont pu être compressées pour limiter l'utilisation de la mémoire tout en captant bien les régions d'intérêt.

4.5.5 Limitations actuelles

Cette expérience, même si elle utilise des données réelles, n'a pas pu être réalisée de manière totalement interactive. Le processus de compression en temps réel est actuellement trop lourd pour être exécuté au rythme avec lequel le capteur fournit les données brutes. Présentement, avec un ordinateur de table standard de type Pentium IV, le système ne peut gérer que 2000 points par secondes. Il faudrait une optimisation d'un facteur 10 pour penser exécuter ce système de manière complètement interactive. Le problème majeur limitant la performance du système est la nécessité, lors de la détermination de la région de recouvrement d'un noeud du niveau n, d'accéder à des noeuds du niveau $n+1$ voisins, qui ne font pas partie des enfants du noeud du niveau n. Atteindre des noeuds voisins dans une structure comme un *octree* demande beaucoup de ressource, car il faut parfois revenir jusqu'à la racine de l'arbre pour les atteindre.



(a) (b) (c)

FIG. 4.10 – Résultats obtenus suite à la reconstruction d'un vase à partir de données réelles. a) État du modèle après l'acquisition du contexte. b) État du modèle suite à l'acquisition d'une région d'intérêt. c) État du modèle suite à l'acquisition d'une seconde région d'intérêt. Les tons de rouge et bleu ont la même signification que ceux utilisés dans la figure 4.5.

Une solution pour contourner ce problème serait de modifier la structure de l'*octree* pour permettre d'atteindre facilement les voisins d'un noeud.

4.6 Conclusion

Nous avons présenté le développement des algorithmes permettant de choisir automatiquement le niveau de résolution optimal auquel reconstruire la surface. Le premier mécanisme permettant de choisir le niveau de résolution local se base sur la densité de points fournie par l'utilisateur et permet la reconstruction de la surface à partir du niveau grossier vers le niveau fin. À cette étape, nous avons défini la notion de *stabilité de la surface*. Une *surface stable* correspond à une surface qui ne devrait pas être grandement influencée lors de l'ajout de nouvelles données. Contrôler la progression des niveaux de résolution permet à l'utilisateur de reconstruire rapidement un contexte en utilisant peu de mémoire. Cependant, le système permet la modélisation de fins détails lorsque la densité locale de points est suffisante.

Le second mécanisme développé, appelé *compression en temps réel,* permet de compresser les zones qui ont été reconstruites à un niveau de résolution trop élevé suite à l'acquisition d'une forte densité de points. Ce mécanisme repose sur le concept de la *condition d'ajustement* qui définit quand la compression temps réel peut être amorcée. Amorcer la compression trop tôt risquerait de produire un modèle manquant de détails et amorcer la compression trop tard serait un gaspillage de mémoire.

Il a également été montré qu'un tel système, en utilisant une stratégie d'acquisition appropriée, permet une réduction notable de la consommation maximale de mémoire en tout temps durant l'acquisition. Cette réduction de la consommation de mémoire permet donc la numérisation d'objets de plus grande taille ou à plus haute résolution pour une même quantité de mémoire. À titre d'exemple, nous avons montré qu'il est possible de reconstruire localement certaines régions détaillées d'un objet à des niveaux de résolution ne pouvant pas être atteints par des méthodes de reconstruction utilisant un niveau de résolution unique.

La compression temps réel nécessite toujours l'intervention de l'utilisateur pour déterminer le niveau de résolution le plus grossier N et le niveau de résolution le plus fin N_{max} devant être utilisés lors de la modélisation de l'objet. Il serait intéressant de développer un mécanisme laissant entièrement au système la possibilité de déterminer automatiquement ces niveaux en fonction des capacités du capteur.

Chapitre 5

Visualisation de la surface

5.1 Introduction

Jusqu'à maintenant, ce document s'est concentré sur les étapes nécessaires à la reconstruction et la compression d'une surface 3D de façon interactive à l'aide d'une représentation vectorielle multirésolution. Très peu d'information a été donnée concernant la dernière étape de la boucle de modélisation interactive, c'est-à-dire la visualisation. Ce choix est justifié par le fait que pour bien comprendre le mécanisme de visualisation de la surface, il faut une bonne compréhension de la manière dont le champ vectoriel est reconstruit. En ayant à l'esprit la structure de données utilisée pour encoder la surface, soit l'*octree,* ainsi que les mécanismes utilisés pour reconstruire une surface multirésolution *cohérente* avec la compression temps réel, la compréhension de la suite de ce chapitre sera facilitée.

Tout d'abord, revenons sur l'utilité de l'étape de visualisation appartenant à la boucle de modélisation interactive. Cette étape vient tout de suite après l'ajout d'une petite quantité de données brutes au modèle et permet à l'utilisateur de voir en temps réel un rendu du modèle 3D en cours de reconstruction. Cette information est essentielle au bon fonctionnement du système de modélisation interactive, car c'est à partir de ce rendu que l'utilisateur sera en mesure de déterminer sa stratégie d'acquisition. En voyant une interprétation en temps réel de la surface en cours de reconstruction, l'utilisateur décide des prochaines régions devant être balayées. Cette première utilité du module de visualisation est de permettre à l'utilisateur de s'assurer que la couverture de l'objet avec les données brutes est complète et suffisante.

De nos jours, permettre de valider la couverture complète de l'objet est insuffisant

pour le module de visualisation. L'interprétation de la surface en temps réel doit être de qualité suffisante pour permettre à l'utilisateur de déterminer s'il y a un problème avec la reconstruction de la surface. L'utilisateur doit être en mesure d'apprécier la qualité de la reconstruction et de réagir en conséquence s'il n'est pas satisfait. Ainsi, il n'est plus nécessaire d'attendre la fin de l'acquisition des données pour détecter une erreur de mise en registre ou un niveau de bruit trop élevé dans les données fournies par le capteur.

Pour permettre à l'utilisateur de valider la couverture de l'objet et la qualité de la reconstruction de la surface, le module de visualisation doit être de qualité et offrir une bonne réactivité. La qualité correspond à la capacité du système à interpréter correctement les données contenues à l'intérieur du champ vectoriel et d'en exploiter le maximum d'information. La réactivité ne prend pas nécessairement la forme d'un affichage à 30 images par seconde comme nous sommes habitués de voir dans les films ou les jeux vidéo. Un système réactif est un système permettant à l'utilisateur de voir le fruit de son travail rapidement, mais pas nécessairement instantanément. Une mise à jour de l'image à toutes les quelques secondes est tout à fait acceptable pour ce genre d'utilisation.

L'approche choisie pour afficher le contenu du champ vectoriel repose sur une technique de lancer de rayons. Comme il a été discuté au chapitre 2, cette approche a été retenue parce qu'elle permet une très haute qualité de rendu. De plus, elle est de complexité constante pour des tailles de volume et d'image fixées. Cela signifie que peu importe la complexité géométrique de la scène, le temps maximal nécessaire au rendu d'une image reste borné et constant, assurant ainsi l'interactivité du système tout au long de la séance d'acquisition. Finalement, un dernier élément penchant en faveur d'un module de visualisation par lancer de rayons est la possibilité d'interpréter les données contenues dans le champ vectoriel directement, sans passer par un représentation alternative telle qu'une liste de polygones. À cause de la nécessité de conserver la réactivité du système, la conversion des données du champ vectoriel vers une représentation alternative pour l'affichage n'est pas envisageable.

De plus, comme le champ vectoriel est encodé dans un *octree*, il contient des données uniquement aux endroits où passe la surface. Implicitement, l'*octree* encode les régions vides de l'espace. Ainsi, une approche par lancer de rayons dans un *octree* peut sauter par-dessus les régions vides et atteindre rapidement la surface. Cette possibilité d'éviter les régions vides est à la base des algorithmes de lancer de rayons rapides [18] et a même été exploitée par des techniques destinées aux processeurs graphiques (GPU) [17]. Comme le montre la suite de ce chapitre, lancer des rayons directement dans le champ vectoriel est une approche avantageuse fournissant une bonne qualité de rendu

en un temps acceptable.

Le chapitre est divisé de la façon suivante. Tout d'abord, le principe du lancer de rayons au travers d'un champ vectoriel sera expliqué en détails pour une représentation monorésolution. Par après, différentes approches d'interprétation des données contenues dans le champ vectoriel seront présentées, toujours pour une représentation monorésolution. L'étape suivante consistera à expliquer le principe permettant le rendu d'une représentation multirésolution. Finalement, comme l'affichage par lancer de rayons reste une étape demandant beaucoup de ressources, une approche fovéale sera présentée pour permettre d'augmenter la réactivité du système.

5.2 Approche par lancer de rayons

Le problème de l'affichage des données contenues dans le champ vectoriel revient à déterminer, en fonction d'un point de vue, quelle partie de la surface apparaît à l'utilisateur et quelle autre partie doit être cachée. Dans un contexte de modélisation interactive, le point de vue correspond généralement à la position du capteur 3D.

Une approche par lancer de rayons solutionne ce problème de façon triviale. Son fonctionnement est montré à la figure 5.1. Pour tous les pixels du plan image, représentés par les rectangles bleus, un rayon partant du centre de projection de la caméra virtuelle et passant par le centre d'un pixel est lancé dans l'espace. Pour déterminer quelle surface doit être affichée à l'écran, il suffit donc de parcourir progressivement le rayon, en partant du centre de projection, jusqu'à ce qu'une première surface soit rencontrée. Dans ce cas-ci, la présence de la surface est déterminée par un passage par zéro de la norme du champ vectoriel. L'intensité du pixel est proportionnelle au produit scalaire du vecteur normal à la surface au point d'impact et du vecteur pointant dans la direction d'observation du rayon. Cette façon de faire simule en fait la loi de Lambert. Si le rayon n'intersecte pas le volume de l'*octree*, le pixel est considéré vide et affiché en noir. Le même principe s'applique pour un rayon traversant le volume de l'*octree* mais ne passant pas au travers de la surface.

Le principe de l'approche par lancer de rayons est fort simple, mais, comme nous le constaterons prochainement, sa mise en oeuvre nécessite plusieurs précautions. La première question que l'on peut se poser est comment fait-on, dans le champ vectoriel, pour déterminer la présence d'une surface ? Théoriquement, il suffit de trouver un passage par zéro de la norme locale du champ vectoriel. Il a été mentionné au chapitre 2 que pour faciliter l'interprétation du modèle, une estimation de la position moyenne

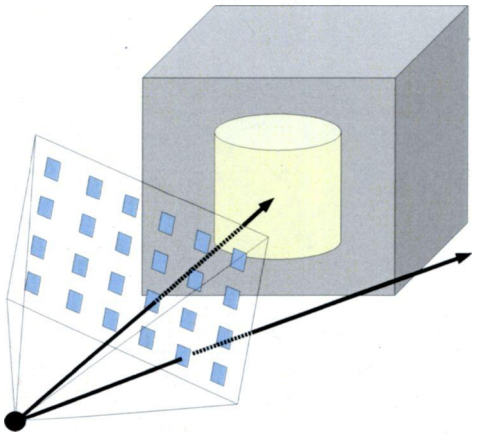


FIG. 5.1 – Fonctionnement de l'approche par lancer de rayons. Pour chaque pixel (rectangles bleus) d'un plan image d'une caméra virtuelle, un rayon partant du centre de projection est lancé dans la direction d'observation. Il suffit donc de suivre ce rayon jusqu'à ce qu'une première surface soit rencontrée où que le rayon quitte les limites du volume modélisé.

du capteur 3D est sauvegardée lors de l'acquisition. La position du capteur 3D permet d'uniformiser le sens des vecteurs normaux à la surface. Cela est fait en s'assurant que les vecteurs normaux pointent toujours dans la direction d'observation du capteur. Ainsi, comme le montre la figure 5.2, les noeuds à l'extérieur de l'objet présentent une distance positive (D+) alors que les noeuds à l'intérieur de l'objet présentent une distance négative (D-). La distance à la surface correspond également à la norme du champ vectoriel. Un passage par zéro de la norme du champ vectoriel revient donc à trouver l'endroit où le rayon passe de l'extérieur de l'objet (norme positive) vers l'intérieur de l'objet (norme négative).

En pratique, trouver la présence de la surface peut se calculer selon différentes approches offrant un compromis au niveau de la performance et de la qualité du rendu. Certaines approches évaluent la position du passage par zéro de la norme du champ vectoriel de manière grossière et d'autres fournissent une estimation plus précise. La prochaine section explique trois techniques différentes permettant de trouver la présence de la surface et en compare les résultats.

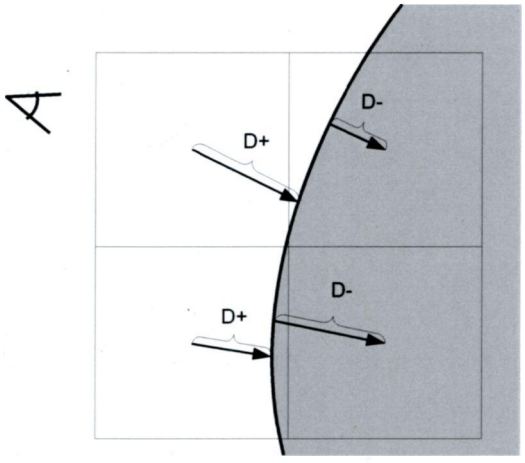


FIG. 5.2 – Uniformisation de l'orientation des vecteurs normaux en fonction de la position du capteur 3D. Les noeuds à l'extérieur de l'objet présentent une distance positive (D+) et les noeuds à l'intérieur de l'objet présentent une distance négative (D-). La distance à la surface correspond également à la norme du champ vectoriel.

5.2.1 Stratégie de détection de la surface

Trois approches différentes ont été étudiées pour détecter la présence de la surface lors de la progression d'un rayon dans le champ vectoriel. Ces trois techniques, de la plus simple à la plus fidèle, sont nommées *premier noeud actif, première intersection valide* et *première entrée* respectivement.

Premier noeud actif

La première technique, appelée *premier noeud actif*, détecte la présence de la surface aussitôt que le rayon lancé entre dans un noeud contenant une section de surface valide dans son volume. En pratique, cela correspond à un noeud dont la distance à la surface la plus près est inférieure à la moitié de sa diagonale, c'est-à-dire une distance de $\frac{\sqrt{3}}{2}$ fois la taille du noeud. La figure 5.3 a) montre le fonctionnement de cette approche. La surface est détectée dès qu'un rayon pénètre dans le volume occupé par un noeud contenant une représentation de la surface à l'intérieur de ses limites. Notez que même si le rayon de gauche, identifié A, semble passer devant la surface encodée, la présence de la surface est détectée. L'avantage indéniable de cette approche réside dans sa simplicité et sa rapidité d'exécution.

Les rendus obtenus avec cette technique sont présentés aux figures 5.3 b) et c) et présentent un très fort effet de discrétisation, surtout sur le pourtour de l'objet. Cet effet de crénelage prend la forme d'un affichage en escalier très déplaisant à observer comme nous pouvons le remarquer sur le profil du nez du mannequin. La raison de ce rendu granuleux est clairement visible dans la figure 5.3 a) : le rayon de gauche (A) est considéré comme traversant la surface même s'il passe devant le plan tangent encodé dans le noeud. L'effet de crénelage provient donc du fait que les frontières des *voxels* sont utilisées pour délimiter les régions de l'espace représentant la surface. Il n'y a aucune interprétation *subvoxel* des données encodées dans le champ vectoriel.

Cette technique est suffisante pour déterminer si la couverture de l'objet est complète, mais elle ne permet pas d'apprécier la qualité de reconstruction de la surface. Pour cette raison, cette approche ne sera pas retenue pour l'implantation finale du module de visualisation.

Première intersection valide

La qualité du rendu peut être grandement améliorée en diminuant le phénomène de crénelage caractéristique de la première approche présentée. L'idée pour améliorer la qualité du rendu est d'exploiter la représentation de la surface encodée en chaque noeud au lieu de n'utiliser que les frontières des *voxels*. Exploiter l'information à un niveau *subvoxel* est une approche fréquemment employée pour augmenter la qualité d'une technique de visualisation basée sur le lancer de rayons. Beaucoup d'approches tentent d'estimer localement la surface par des splines de différents degrés [29, 24]. Malheureusement, ces approches requérent l'évaluation du champ vectoriel dans un voisinage de taille non négligeable et cela demande trop de ressources processeur pour le contexte actuel. Nous nous limiterons donc à l'utilisation des plans tangents déjà présents dans les noeuds.

La seconde approche présentée, appelée *première intersection valide*, exploite l'information encodée en chaque noeud en détectant la présence de la surface lorsqu'il y a une intersection entre le rayon en provenance de la caméra virtuelle et un plan tangent valide. En pratique, chaque plan tangent a un support fini et représente une section locale de la surface. Le support du plan tangent correspond à un cercle de rayon $\frac{\sqrt{3}}{2}$ fois la taille d'un *voxel*. La figure 5.3 d) présente le principe de fonctionnement de cette approche et les figures 5.3 e) et f) montrent l'amélioration notable au niveau de la qualité du rendu malgré une complexité de calcul accrue de l'ordre de 20%. Notez que dorénavant, le rayon A n'intersecte plus la surface comme avec la méthode *premier noeud actif* , d'où une amélioration de la qualité nettement visible dans le pourtour des

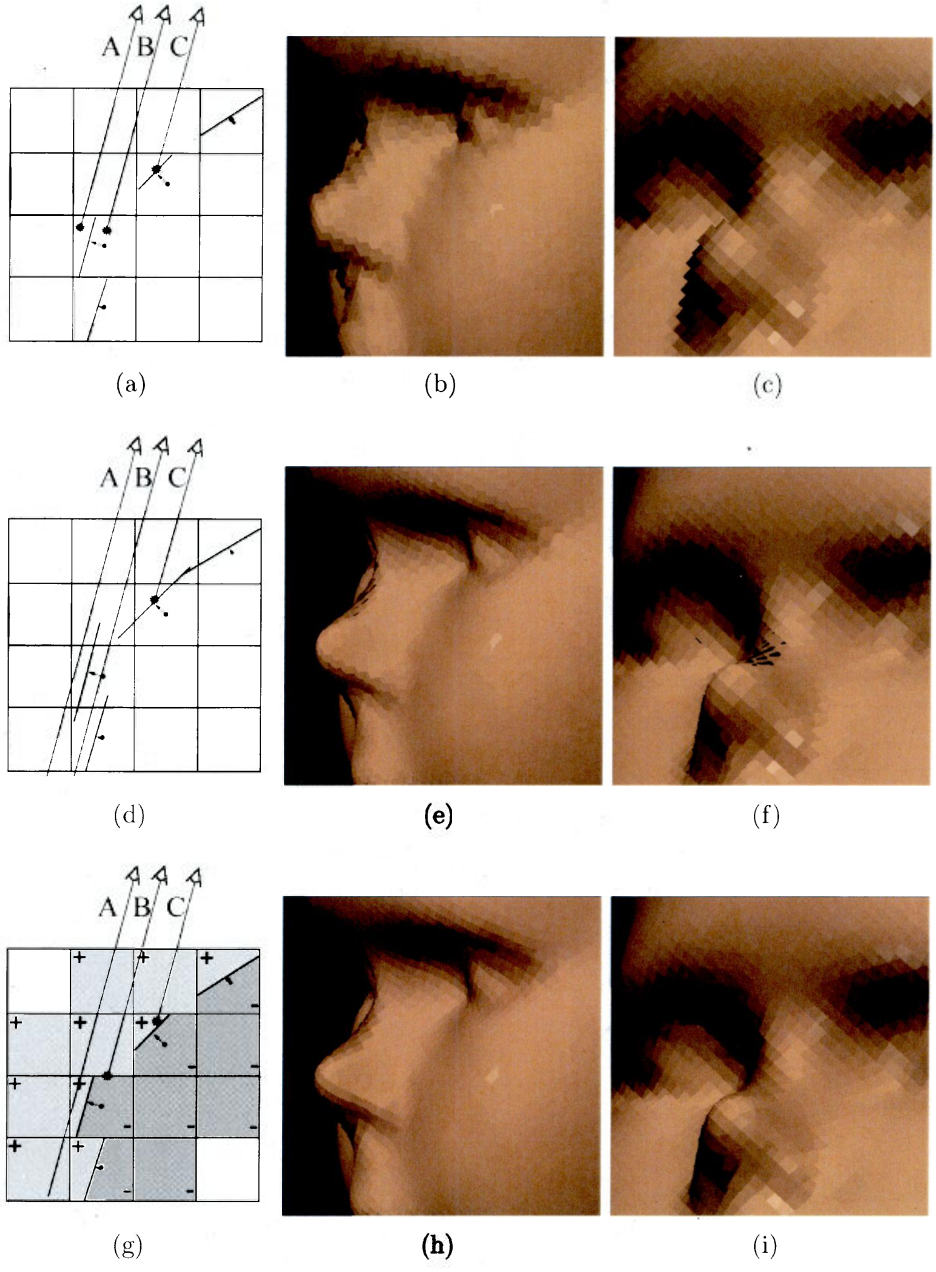


FIG. 5.3 – Comparaison des trois différentes techniques de détection de la présence d'une surface. a)-c) *premier noeud actif.* d)-f) *première intersection valide.* g)-i) *première entrée.* La colonne 1 montre un schéma illustrant le fonctionnement de la technique. Les colonnes 2 et 3 montrent des rendus du modèle monorésolution de la tête du mannequin pour chaque technique.

yeux et du nez de la tête du mannequin.

Malheureusement, cette seconde approche introduit un nouveau type d'artéfact de visualisation visible sur le dessus du nez du mannequin dans la figure 5.3 f). Certains rayons passent au travers de l'*octree* sans jamais rencontrer la surface. Cela introduit des pixels noirs dans l'image prenant l'apparence de trous dans la surface.

En pratique, ces artéfacts sont dus au fait que certains rayons passent au travers de la surface par de faux tunnels. Aucune intersection directe entre le rayon et les sections de surface encodées dans les noeuds du champ vectoriel n'a lieu et le module de visualisation considère qu'aucune surface n'est présente à cet endroit en affichant un pixel noir. Le rayon B de la figure 5.3 d) montre clairement cette situation fort déplaisante. Ce phénomène est d'autant plus présent dans les régions où la normale à la surface est presque perpendiculaire avec le rayon en provenance de la caméra virtuelle.

Il serait possible de remédier à ce problème en augmentant le support de chacune des sections locales de la surface. Cette solution est plus ou moins intéressante, car elle produirait un rendu plus grossier de l'objet. Une autre technique serait d'augmenter la taille de l'enveloppe de reconstruction. De cette manière, il serait plus difficile pour un rayon de passer au travers du volume sans entrer en contact avec une section de la surface. Cette approche a le désavantage d'influencer le processus de reconstruction. Comme il a été précédemment mentionné au chapitre 2, une enveloppe de reconstruction de grande taille a pour effet de reconstruire un modèle plus lisse, contenant moins de détail. Au lieu de modifier la structure du champ, il serait possible de régler le problème des faux tunnels en lançant dans le volume des rayons ayant un diamètre proportionnel à la taille des pixels du plan image. Cette approche, appelée en anglais *Beam Tracing*, a l'avantage d'éviter les faux tunnels tout en fournissant un rendu présentant moins de crénelage [10, 12]. Encore une fois, cette approche a été mise de côté à cause du surplus de calculs demandé.

La solution retenue pour pallier à ce problème est d'exploiter directement la représentation vectorielle en faisant une recherche progressive, le long du rayon, du passage par zéro de la norme du champ vectoriel pour trouver la présence de la surface. Cette troisième approche porte le nom de *première entrée* et est expliquée en détails dans la section suivante.

Première entrée

Une solution intéressante au problème des faux tunnels, n'amenant pas de surcharge de calculs importante, est donc d'exploiter les avantages de la représentation volumétrique dans laquelle la surface est définie, à l'intérieur des limites de l'enveloppe de reconstruction. Cette approche combine les avantages des deux précédentes tout en évitant les problèmes de visualisation précédemment présentés.

L'idée derrière cette technique est de déterminer, pour chaque noeud encodant une section de surface valide, si le noeud est à l'intérieur de l'objet (signe -), à l'extérieur de l'objet (signe +) ou à l'interface de la surface de l'objet (signe + et -). Un noeud étant à la fois à l'intérieur et à l'extérieur de l'objet est un noeud divisé en deux par sa propre section de surface. Déterminer si un noeud est à l'intérieur ou à l'extérieur de l'objet se fait simplement en évaluant la distance du noeud à la surface.

L'approche par lancer de rayons utilise donc la position relative du noeud, soit à l'intérieur ou à l'extérieur de l'objet, pour produire un rendu cohérent du modèle 3D. La stratégie utilisée consiste donc à chercher un passage par zéro de la norme du champ vectoriel le long du rayon lancé. Pour ce faire, il suffit de parcourir progressivement le rayon, à partir du centre de projection de la caméra virtuelle, et d'identifier un changement de signe de la norme du champ vectoriel. Le secret de cette approche est de vérifier à intervalle régulier le signe du champ vectoriel le long du rayon pour éviter des artéfacts du genre faux tunnels. Il faut faire attention à ne pas utiliser un intervalle trop petit pour ne pas alourdir inutilement les calculs.

Une observation attentive de la structure du champ vectoriel permet de déterminer qu'un changement de signe ne peut se produire que dans deux situations distinctes :

1. Lorsqu'un plan tangent croise explicitement le rayon,
2. Lorsque le rayon entre dans un nouveau *voxel*, c'est-à-dire à l'intersection entre deux *voxels*.

L'utilisation de l'information contenue dans le champ vectoriel en exploitant les intersections directes entre le plan tangent encodé en chaque noeud et le rayon assure un rendu de qualité avec peu d'effet de crénelage. Les faux tunnels sont évités en vérifiant également, à chaque intersection entre deux *voxels*, qu'un changement de signe de la norme du champ vectoriel n'a pas eu lieu. S'il y a un changement de signe malgré le fait qu'aucun plan tangent n'ait été rencontré explicitement, le système déclare qu'une surface est présente à cet endroit. La figure 5.3 g) présente bien le comportement de

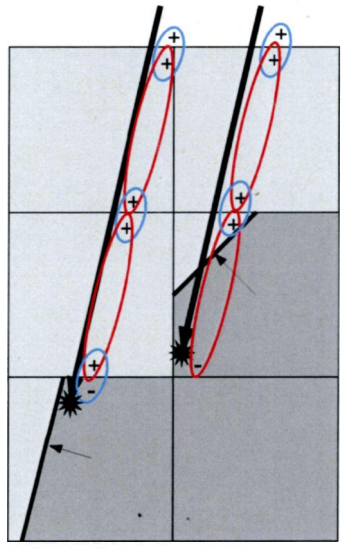


FIG. 5.4 – Positions d'évaluation de la norme du champ vectoriel lors de la progression d'un rayon au travers du volume. Un cercle rouge correspond à une évaluation d'un passage par zéro à l'intérieur des limites d'un *voxel*. Un cercle bleu correspond à une évaluation d'un passage par zéro à l'intersection de deux *voxels*.

cette approche. Le rayon A réagit comme il se doit en ne détectant pas la présence de la surface. La figure 5.3 h) montre effectivement un contour lisse du nez du mannequin. En plus, comme le montre le rayon B, la détection d'un passage par zéro à chaque intersection de *voxels* évite le phénomène de faux tunnels. Les trous dans l'objet visibles lors de l'utilisation de l'approche *première intersection valide* ne sont plus visibles dans les figures 5.3 h) et i). Il est désormais possible d'obtenir un rendu cohérent de la surface encodée dans le champ vectoriel tout en évitant une trop forte discrétisation ainsi que les faux tunnels. Le gain important obtenu au niveau de la qualité justifie amplement la perte de 40% au niveau performance par rapport à la première approche présentée.

La figure 5.4 montre en détails les positions auxquelles la norme du champ vectoriel doit être évaluée pour détecter la présence d'une surface. Pour détecter la présence de la surface à l'intérieur des limites d'un *voxel*, il suffit de comparer le signe juste après être entré dans le *voxel* et tout juste avant d'en sortir. Un changement de signe indique la présence d'une surface. Ces cas sont entourés en rouge dans la figure 5.4. Pour déterminer la présence d'une surface à l'intersection de deux *voxels*, il suffit d'évaluer le signe avant de quitter le premier *voxel* et tout juste après l'entrée dans le second. Ces cas sont entourés en bleu dans la même figure. Dans cet exemple, le rayon de gauche présente une intersection implicite, aux frontières entre deux *voxels*, et le rayon de droite

présente une intersection directe avec la surface.

5.2.2 Stratégie de parcours du volume

L'approche *première entrée* offre un rendu de haute qualité tout en évitant des artéfacts de visualisation déplaisants. Cependant, cette technique nécessite de connaître exactement le point d'entrée et de sortie du rayon pour déterminer si ce dernier est à l'intérieur ou à l'extérieur de l'objet. Pour le lancer d'un simple rayon, cette évaluation peut se faire plusieurs dizaines de fois, d'où la nécessité d'optimiser cette étape de calcul.

Déterminer les points d'entrée et de sortie exacts du rayon pour un *voxel* peut se faire en calculant l'intersection entre le rayon et les six plans délimitant ses frontières. Les deux intersections à l'intérieur des limites des faces du *voxel* correspondent au point d'entrée et au point de sortie du rayon. Cette approche, quoique valable, demande beaucoup de calculs et n'est pas appropriée à la présente tâche. Une autre approche, développée par Revelles *et al.* [23], se base sur la représentation paramétrique d'un rayon pour accélérer grandement l'estimation, pour chaque *voxel,* des points d'entrée et de sortie du rayon. Cet algorithme a été adapté pour permettre son utilisation dans le module de visualisation. En voici le principe de fonctionnement pour la traversée d'un *octree.*

Traversée paramétrique d'un *octree*

Un rayon sous la forme paramétrique est représenté par un point de départ p_0 et un vecteur direction unitaire $\overrightarrow{d}$. Pour tout λ_i réel et positif, l'expression suivante correspond à un point appartenant au rayon :

$$\lambda_i \overrightarrow{d} + p_0 = p_i \tag{5.1}$$

Il est donc possible d'identifier n'importe quel point appartenant à ce rayon en sauvegardant uniquement le paramètre λ_i. En considérant le problème sous un autre angle, il est facile de trouver le paramètre λ donnant la position d'intersection du rayon avec un plan selon les axes du système de coordonnées :

$$\lambda_x = (X - p_{0,x})/\overrightarrow{d_x} \tag{5.2}$$

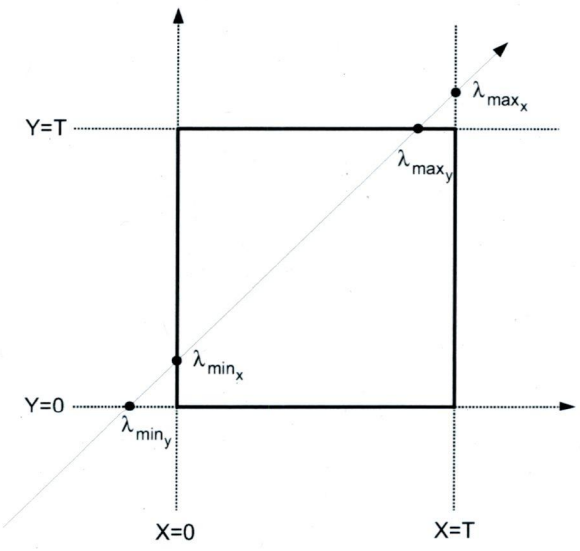


FIG. 5.5 – Intersections entre les plans délimitant le volume de l'*octree* et le rayon lancé. Par souci de simplicité, seule la vue présentant les axes X et Y est affichée. Le point d'entrée dans l'*octree* correspond à $max(\lambda_{min_x}, \lambda_{min_y}, \lambda_{min_z})$. Le point de sortie correspond à $min(\lambda_{max_x}, \lambda_{max_y}, \lambda_{max_z})$.

$$\lambda_y = (Y - p_{0,y})/\overrightarrow{d_y} \tag{5.3}$$

$$\lambda_z = (Z - p_{0,z})/d_z \tag{5.4}$$

Les paramètres, sur le rayon, des points d'intersection avec les plans $x = X$, $y = Y$ et $z = Z$ sont respectivement λ_x, λ_y et λ_z. Comme le montre la figure 5.5, il est donc possible de connaître les points d'intersection entre le rayon et les plans délimitant le volume de l'*octree*. En positionnant un *octree* de taille T à l'origine du système de coordonnées, ces plans correspondent à $x = 0$, $y = 0$, $z = 0$, $x = T$, $y = T$, et $z = T$. Une fois ces six valeurs de λ obtenues, la suite de l'algorithme n'utilise que des comparaisons, des additions ainsi qu'une division par 2 pour trouver les points d'entrée et de sortie des noeuds de l'*octree*.

Une observation attentive de la figure 5.5 montre que trois valeurs de λ sont situées près de l'origine du rayon lancé, notées λ_{min} et les trois autres sont situées à l'extrémité du rayon, une fois que ce dernier a traversé le volume de l'*octree*. Ces dernières valeurs sont dénotées λ_{max}. Pour déterminer le point d'entrée du rayon dans le noeud racine, il suffit de prendre la valeur maximale des λ_{min}. Le point de sortie correspond au minimum des trois valeurs λ_{max} :

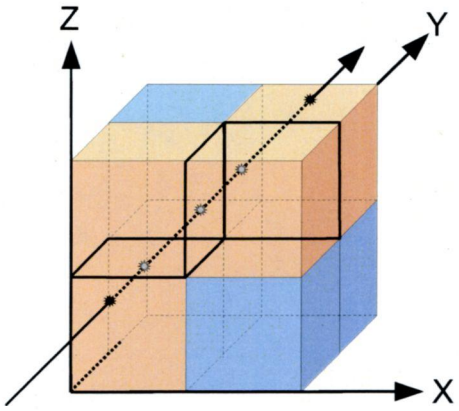


FIG. 5.6 – Le nombre maximum d'octants enfants parcourus lors de la traversée d'un noeud parent par un rayon est de 4. Pour bien comprendre l'image, il faut voir que le rayon traverse trois octants dans le premier groupe orienté selon le plan XZ et sort par un octant appartenant au deuxième groupe orienté selon le même plan, mais situé à une position Y supérieure.

$$\lambda_{entree} = max(\lambda_{min_x}, \lambda_{min_y}, \lambda_{min_z}) \quad (5.5)$$

$$\lambda_{sortie} = min(\lambda_{max_x}, \lambda_{max_y}, \lambda_{max_z}) \quad (5.6)$$

Si λ_{entree} est plus grand que λ_{sortie}, le rayon ne passe tout simplement pas au travers du volume défini par l'*octree*. Il est important de noter que $\lambda_{min_x} = \lambda_{x=0}$, $\lambda_{min_y} = \lambda_{y=0}$, $\lambda_{min_z} = \lambda_{z=0}$ et que $\lambda_{max_x} = \lambda_{x=T}$, $\lambda_{max_y} = \lambda_{y=T}$, $\lambda_{max_z} = \lambda_{z=T}$ seulement si les composantes du vecteur direction du rayon sont positives. Si une des composantes est négative, cela signifie que pour cet axe, le rayon entrera par le plan passant par la valeur T et sortira par le plan passant par l'origine du système de coordonnées, soit $\lambda_{min} = \lambda_T$ et $\lambda_{max} = \lambda_0$. Donc, pour le calcul des λ_{min} et λ_{max} initiaux, il faut tenir compte des signes des composantes du vecteur direction et choisir les paramètres en conséquence.

Une fois que les points d'entrée et de sortie sont déterminés, il faut trouver la liste des octants enfants que traverse le rayon avant de quitter le noeud parent. L'ordre dans lequel les octants sont visités est également très important pour assurer le bon fonctionnement du lancer de rayons. Une approche par recherche exhaustive de voisins ne fournirait pas implicitement l'ordre de parcours des octants enfants et consommerait plus de puissance de calcul qu'il n'en faut, car le nombre maximum d'octants enfants visités, dans le pire cas, n'est que de 4. La figure 5.6 montre la pire situation à laquelle le

système doit faire face avant de quitter le noeud parent. Cette situation arrive lorsque, suite à la visite du premier octant enfant, le rayon visite trois autres octants chacun correspondant à une position différente selon chacun des axes.

Cette petite mise en situation montre qu'il est préférable d'utiliser une approche plus raffinée pour trouver la liste des octants enfants traversés par le rayon. La solution proposée repose sur la détermination de la face par laquelle le rayon sort d'un octant enfant. Ainsi, en sachant par quelle face le rayon quitte un octant, le prochain octant visité est déterminé automatiquement. La question est donc de déterminer par quelle face le rayon quitte un octant enfant. C'est ici que l'algorithme basé sur la représentation paramétrique du rayon montre tout son intérêt. Comme le montre la figure 5.7, il suffit de calculer les paramètres intermédiaires (λ'_x, λ'_y, λ'_z) et de les classer en ordre croissant tout en ne gardant que les paramètres appartenant à l'intervalle $[\lambda_{entree}, \lambda_{sortie}]$. Les paramètres à l'extérieur de cet intervalle ne doivent pas être tenus en compte, car ils représentent des points de contact à l'extérieur du volume du noeud racine.

Si le plus petit des trois paramètres intermédiaires est λ'_x, la face de sortie correspond donc au plan YZ. Au contraire, si la plus petite valeur est λ'_y, la face de sortie est le plan XZ. Dans le cas où λ'_z est la plus petite valeur, la face de sortie est le plan XY. Cela signifie que par un simple ordonnancement des paramètres intermédiaires, il est possible de trouver la liste ordonnée des octants enfants traversés par le rayon.

Une observation attentive de la figure 5.7 montre géométriquement que les calculs des paramètres intermédiaires sont des plus simples. Il suffit de moyenner les couples de paramètres maximum et minimum obtenus pour le noeud racine :

$$\lambda'_x = \frac{\lambda_{min_x} + \lambda_{max_x}}{2} \tag{5.7}$$

$$\lambda'_y = \frac{\lambda_{min_y} + \lambda_{max_y}}{2} \tag{5.8}$$

$$\lambda'_z = \frac{\lambda_{min_z} + \lambda_{max_z}}{2} \tag{5.9}$$

Cela signifie que les paramètres intermédiaires sont à mi-chemin entre les paramètres obtenus pour le noeud racine et ce, pour chacun des axes. En plus de permettre de déterminer la liste des octants enfants traversés par le rayon, la liste des paramètres intermédiaires permet de calculer la position 3D des points de sortie et d'entrée des octants enfants. Notons que ces positions 3D sont utilisées par l'algorithme de détection de la surface *première entrée* et que l'utilisation des paramètres intermédiaires facilite

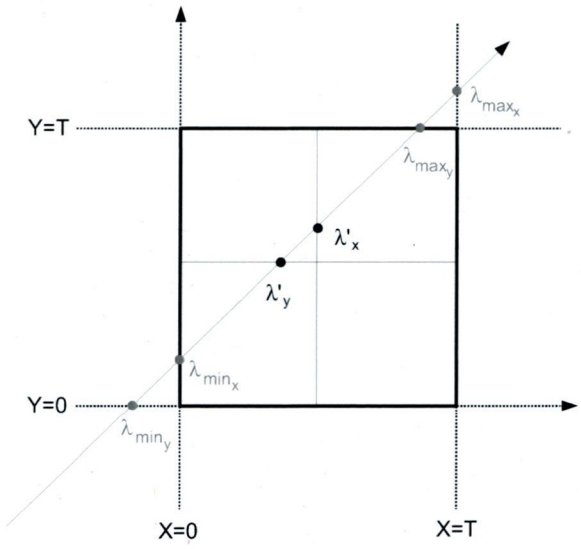


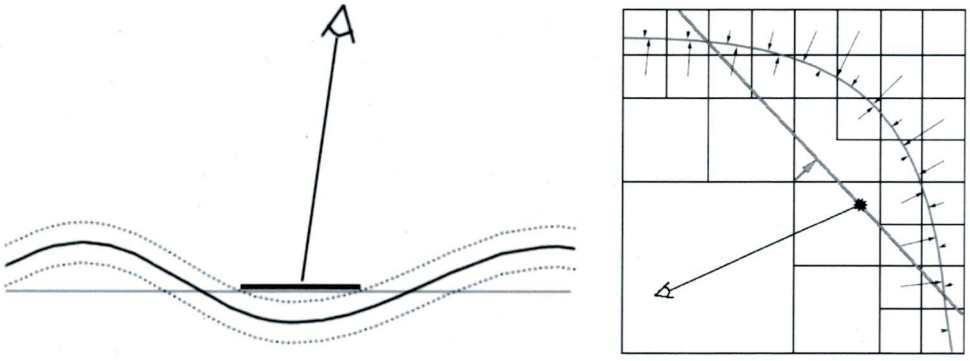
FIG. 5.7 – Paramètres intermédiaires (λ'_x, λ'_y, λ'_z) permettant de déterminer les faces des octants enfants par lesquelles le rayon passe. Par souci de simplicité, uniquement les axes X et Y sont présentés.

de beaucoup leur calcul.

Une fois que les noeuds enfants traversés ont été déterminés pour un niveau de résolution, il faut appliquer l'algorithme récursivement à ces noeuds enfants pour trouver la liste de tous les noeuds traversés par le rayon. Les détails d'une implantation efficace de cette approche sont présentés en annexe.

Adaptation à la multirésolution

Il est maintenant possible, grâce à l'algorithme précédemment expliqué, de parcourir tous les noeuds de l'*octree* qui croisent un rayon et ce, de manière progressive en partant de ses frontières. Comme il a été mentionné au chapitre 3, le champ vectoriel multirésolution est composé de plusieurs versions locales de la surface, à des niveaux de résolution différents. Il faut donc que le module de visualisation interprète correctement le contenu du champ. Pour ce faire, le lancer de rayons utilise la règle d'interprétation précédemment mentionnée : en chaque point du volume, toujours utiliser la représentation valide qui est reconstruite au niveau de résolution le plus fin localement. Cette simple règle assure donc que si une représentation haute résolution est disponible à un certain endroit dans le volume, elle sera utilisée pour déterminer si le



(a) (b)

FIG. 5.8 – Importance de la *continuité des niveaux de résolution* lors de la reconstruction de la surface multirésolution pour permettre un affichage cohérent. a) Pour une approche par lancer de rayons, une représentation très grossière peut cacher une représentation plus fine de la même surface. b) Comme la *continuité des niveaux de résolution* n'a pas été respectée, la représentation au niveau grossier N cache la résolution fine encodée au niveau fin $N + 3$.

rayon a pénétré à l'intérieur de l'objet ou s'il est encore à l'extérieur. Cette approche ne fournira un rendu valide que si la *continuité des niveaux de résolution* est respectée. Nous allons maintenant expliquer pourquoi la *continuité des niveaux de résolution* est nécessaire à l'interprétation cohérente du champ vectoriel multirésolution à l'aide d'une approche par lancer de rayons.

Rappelons que le respect de la *continuité des niveaux de résolution* résout le problème de la présence d'une représentation double de la surface dans les endroits contenant beaucoup de détails ou de fortes courbures (section 3.4.2). Reconstruire les niveaux de résolution intermédiaires permet donc d'assurer que les sections de surface encodées aux niveaux grossiers soient recouvertes par des noeuds valides de résolution plus fine. Cette condition est respectée, rappelons-le, seulement si l'erreur commise par la représentation à chaque niveau de résolution est inférieure à la taille de l'enveloppe de reconstruction du niveau suivant (voir le chapitre 3 pour les détails). La figure 5.8 a) montre, dans le cas d'une surface sinusoïdale, le genre d'erreur d'affichage généré lorsque les niveaux intermédiaires ne sont pas reconstruits. À certains endroits, soit les crêtes du sinus dans ce cas-ci, la première surface valide rencontrée est celle encodée au niveau le plus fin. Cependant, dans les creux du sinus, la première surface rencontrée est celle encodée au niveau grossier. Dans cette situation, il y aurait donc certaines sections de la surface affichées au niveau le plus fin et d'autres affichées au niveau le plus grossier et ce, même si le niveau fin est valide sur la totalité de la surface.

La figure 5.8 b) montre un problème similaire à l'aide d'un représentation schématique

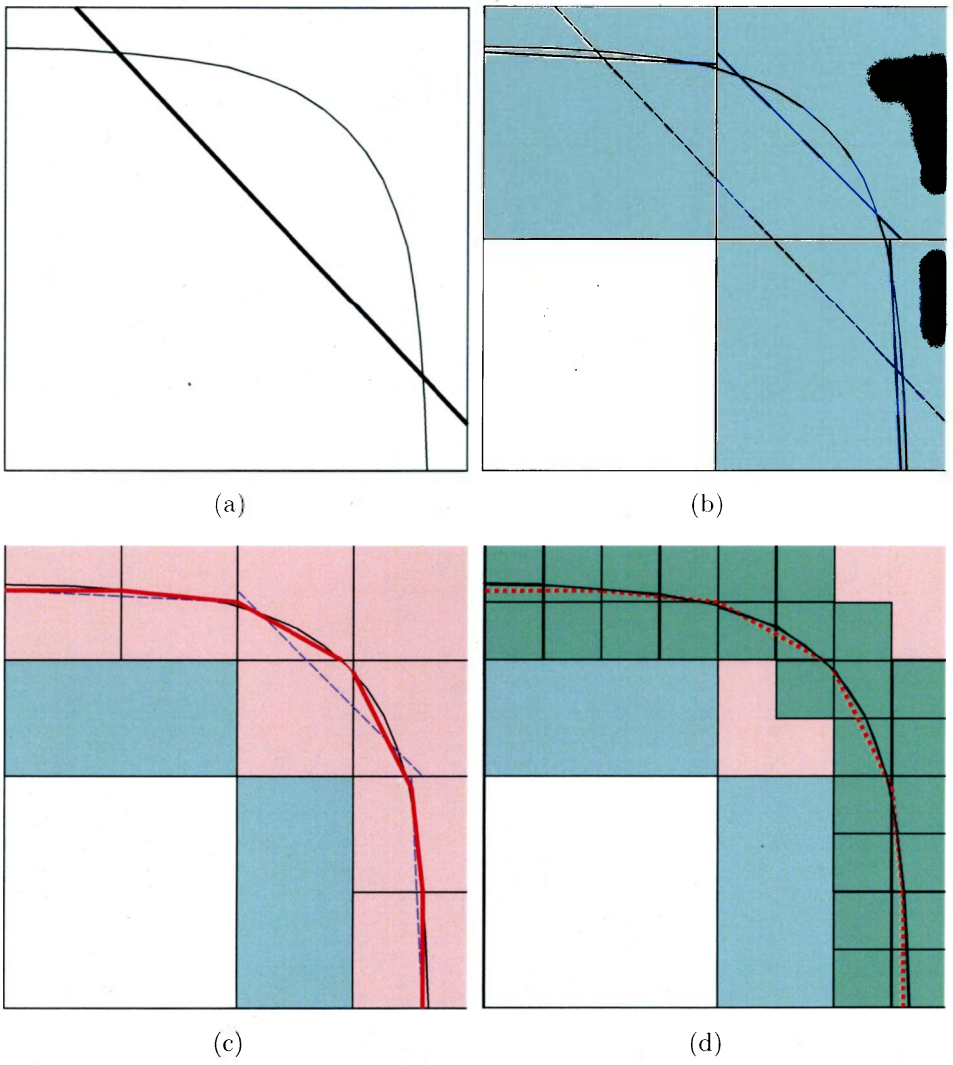


FIG. 5.9 – Problème d'affichage réglé par la *continuité des niveaux de résolution*. a) Représentation au niveau grossier N. b) Lors de la reconstruction du niveau N+1, les noeuds valides recouvrent et désactivent la représentation au niveau N maintenant en pointillés. c) Le même phénomène a lieu lors de la reconstruction du niveau N+2. d) La reconstruction du niveau N+3 désactive en totalité la surface encodée au niveau N+2 et permet l'affichage cohérent de la surface multirésolution au niveau le plus fin disponible. Les niveux N et N+1 ont bel et bien été désactivés progressivement.

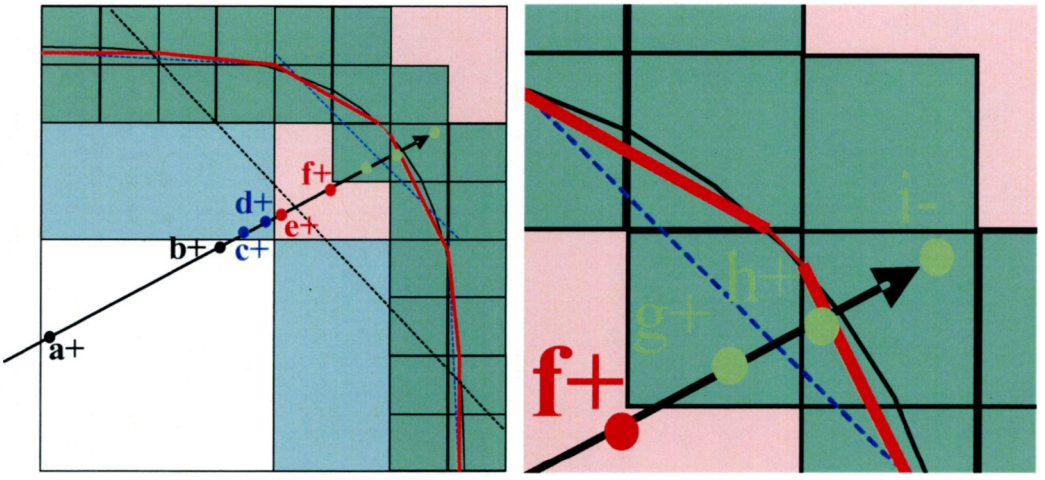


FIG. 5.10 – a) Progression d'un rayon dans un champ vectoriel multirésolution respectant la *continuité des niveaux de résolution*. b) Zoom sur la partie entourant le point d'intersection entre le rayon et la surface. Voir le texte pour les explications.

de l'*octree*. Lorsque la *continuité des niveaux de résolution* n'est pas respectée, la représentation grossière du niveau N n'a pas la possibilité de se faire désactiver par les noeuds valides du niveau $N + 3$. Le rayon frappe donc le niveau grossier avant d'avoir atteint le niveau fin. Tel que montré à la figure 5.9, la *continuité des niveaux de résolution* permet d'éviter ce problème. Tout d'abord, le niveau grossier N est reconstruit (figure 5.9 a)). La reconstruction du niveau N+1 permet de recouvrir la représentation du niveau N avec des noeuds valides, ce qui a pour effet de désactiver cette représentation grossière maintenant affichée en pointillés dans la figure 5.9 b). Le même phénomène est visible lors de la reconstruction du niveau N+2 (figure 5.9 c)). Lorsque ce niveau est reconstruit, nous constatons que le niveau fin N+3 a la possibilité de recouvrir en totalité les données encodées au niveau N+2 (figure 5.9 d)). Lors de l'affichage, c'est effectivement la représentation reconstruite au niveau le plus fin qui sera affichée.

Les figures 5.10 a) et b) montrent un exemple de la progression d'un rayon dans le champ vectoriel précédemment étudié. La figure a) montre le volume dans son ensemble et la figure b) montre un zoom de la région entourant le point d'intersection avec la surface. À première vue, l'image peut sembler complexe, mais elle montre bien le principe de fonctionnement de l'algorithme de visualisation. Chaque point sur le rayon représente un endroit où le champ vectoriel a dû être évalué pour déterminer si le rayon est à l'extérieur ou à l'intérieur de l'objet. La couleur du point indique le niveau de résolution le plus fin disponible à cet endroit. Voici maintenant les explications pour chacun de ces points d'évaluation :

- a) À cet endroit, le rayon pénètre dans le volume étudié. Seul le niveau le plus grossier N est disponible. La valeur lue dans le champ indique que le rayon est présentement à l'extérieur de l'objet.
- b) Juste avant d'entrer dans un octant enfant valide, le système vérifie s'il n'y a pas eu d'intersection explicite avec la surface au niveau N. Le point 3D est calculé et une vérification est faite pour déterminer s'il est à l'intérieur ou à l'extérieur de l'objet par rapport à la section de surface encodée.
- c) Lorsque le rayon pénètre dans le premier noeud enfant, une autre vérification est faite pour déterminer s'il y a eu un passage par zéro implicite. À cet endroit, le niveau de résolution local le plus fin disponible n'est pas le niveau N, mais bien le niveau N+1 (en bleu). Le rayon est toujours à l'extérieur de l'objet.
- d) Vérification d'une intersection explicite. Le niveau de résolution disponible le plus fin à cet endroit est encore N+1.
- e) La vérification implicite à cet endroit est faite avec les représentations encodées aux niveaux N+1 et N+2. Notez que la représentation au niveau N est complètement désactivée et qu'elle n'intervient pas lors de la prise de décision.
- f) Le rayon, avant de quitter le *voxel* du niveau N+2, passe par un *voxel* du niveau N+3. Avant d'entrer dans le *voxel* du niveau N+3, une vérification d'intersection explicite doit être faite au niveau N+2.
- g) La frontière entre le niveau N+2 et le niveau N+3 ne présente toujours pas de passage par zéro. À cette étape, il y a également la vérification d'une intersection explicite. Ici, la figure est trop petite pour afficher les points d'entrée et de sortie. Un seul point est présent pour les deux.
- h) Une autre vérification d'un passage par zéro implicite entre les représentations au niveau N+3.
- i) Détection d'un passage par zéro à l'intérieur d'un *voxel* du niveau N+3. Le rayon s'arrête ici.

Cet exemple montre que grâce au respect de la *continuité des niveaux de résolution*, la représentation affichée est bel et bien celle encodée au niveau de résolution le plus fin disponible. Comme la représentation du niveau grossier a été désactivée, elle n'intervient pas lors de l'interprétation des données et une image cohérente peut être affichée à l'utilisateur.

Après avoir détecté le passage par zéro, rappelons que l'intensité du pixel affiché est déterminée en fonction du cosinus de l'angle entre la normale à la surface et la direction du rayon lancé. Pour améliorer la qualité du rendu, il est possible d'utiliser un modèle de réflectance tel que *Phong [22]* et de gérer différentes sources lumineuses. La figure 5.11 montre le rendu d'un objet réel, soit le vase utilisé au chapitre précédent, tout en limitant le niveau de résolution maximal pouvant être affiché. En a), seul le niveau

(a) (b)

(c) (d)

(e)

FIG. 5.11 – Rendus d'un modèle multirésolution obtenus à partir d'un objet réel, soit le vase utilisé au chapitre précédent. a), b), c), d) Le niveau de résolution maximal est limité à 8, 9, 10 et 11 respectivement. e) Même vue qu'en d) mais en affichant les niveaux de résolutions en différentes couleurs. Le niveau grossier est affiché en gris, les niveaux intermédiaires 9 et 10 en teintes de rouge et en teintes de bleu respectivement et le niveau fin en beige.

le plus grossier, soit N=8, est visible. Les figures b), c) et d) montrent le modèle en fixant le niveau de résolution maximal à 9, 10 et 11 respectivement. La figure e) montre le résultat final en permettant au système d'afficher le niveau de résolution le plus fin contenu dans la représentation vectorielle. Une observation attentive du rendu haute résolution du modèle montre un effet de flou dans le centre de la fleur. Il est possible de distinguer des traits diagonaux dans l'objet. Ce phénomène n'est pas lié au module de visualisation, mais plutôt à une mauvaise calibration du capteur 3D lors de la prise des données. Cet exemple montre bien que le module de visualisation est de qualité suffisante pour évaluer la reconstruction de la surface durant l'acquisition des données. Dans la figure 5.11 e), le modèle est affiché en utilisant des teintes de gris pour le niveau grossier, des teintes de rouge pour le niveau 9, des teintes de bleu pour le niveau 10 et du beige pour le niveau le plus fin.

5.3 Approche fovéale

Malgré le fait que l'approche par lancer de rayons soit de complexité constante pour une taille d'image fixe et un niveau de résolution maximal donné, il n'en reste pas moins que cette étape est la plus gourmande de toutes celles faisant partie de la boucle de modélisation interactive. Pour diminuer la consommation de ressources processeur tout en fournissant un rendu interactif à l'usager, il aurait été envisageable de développer une version du lancer de rayons fonctionnant sur un processeur graphique. Ces approches donnent de bons gains de performance [17, 2]. Cependant, elles sont difficilement envisageables dans le contexte de la modélisation interactive à cause de la taille de la structure de données utilisée. En chaque noeud, plus de 68 octets sont nécessaires pour encoder la surface. Il n'est pas rare qu'un modèle utilise plus de 512 Mo et même plus de 1024 Mo. Cette quantité de mémoire n'est pas encore disponible sur les cartes vidéo actuelles.

L'approche choisie pour améliorer la réactivité du système repose sur un affichage fovéal. Cette approche consiste, lors de l'affichage d'une nouvelle image, à concentrer les rayons lancés dans la zone centrale, zone contenant habituellement les détails d'intérêt pour l'utilisateur. En pratique, l'image à afficher est divisée en trois régions concentriques, présentées à la figure 5.12 a). Les régions ont respectivement une dimension de 128x128 pixels, 256x256 pixels et 512x512 pixels. Initialement, les pixels appartenant à chaque région sont regroupés pour former des ensembles de 4x4 pixels dans la région centrale, 8x8 pixels dans la région intermédiaire et 16x16 pixels dans la région extérieure.

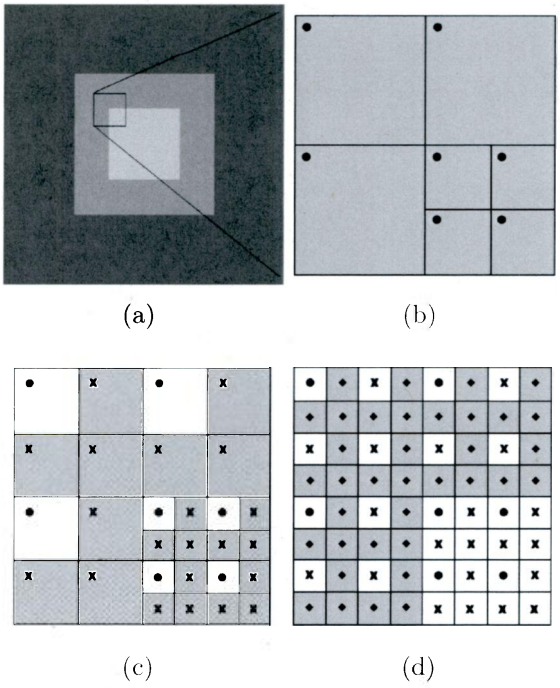


FIG. 5.12 – a) L'approche fovéale divise l'image en trois régions concentriques de 128x128 pixels, 256x256 pixels et 512x512 pixels. b-d) Phases 2, 3 et 4 de l'approche fovéale montrant le coin supérieur gauche de la frontière entre la zone centrale et la zone intermédiaire.

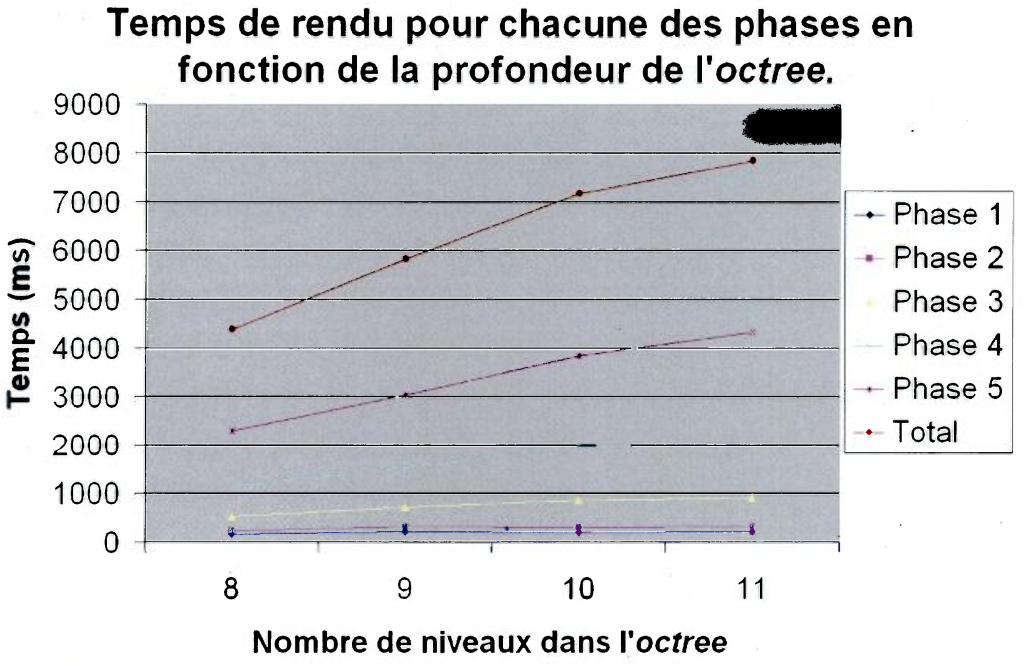


FIG. 5.13 – Temps de rendu nécessaire pour les différentes phases de l'approche fovéale en fonction de la profondeur maximale de l'*octree*.

Lorsque le rendu d'une image débute, un rayon par groupe de pixels est lancé au travers du volume. Cela signifie que 1 pixel sur 16 est affiché dans la région centrale, 1 pixel sur 64 est affiché dans la région intermédiaire et 1 pixel sur 256 est affiché dans le pourtour de l'image. Au total, seulement 2560 rayons sur un potentiel de 262144 rayons (512x512) sont lancés pour fournir à l'utilisateur un aperçu de la surface en cours de reconstruction. Cet aperçu, quoique très grossier, est suffisant pour guider l'utilisateur dans sa démarche d'acquisition. La figure 5.13, montrant les temps de rendu d'une image de 512x512 pixels pour les différentes phases en fonction de la profondeur de l'*octree*, confirme que ce premier aperçu peut être obtenu environ six fois par seconde avec un ordinateur de table standard, fréquence acceptable pour conserver l'interactivité du système.

La seconde phase de l'approche fovéale consiste à subdiviser, pour chaque zone, chacun des groupes de pixels et de lancer un rayon pour chaque nouveau sous-groupe de pixels ainsi formé. La figure 5.12 b) montre un grossissement du coin supérieur gauche de la frontière entre la zone centrale et la zone intermédiaire pour cette phase. Notez que maintenant, la zone centrale est formée de groupes de 2x2 pixels et la zone intermédiaire comprend des groupes de 4x4 pixels. Cette figure montre également que pour chaque groupe, le rayon lancé passe par le pixel situé dans le coin supérieur gauche. Une fois

la surface détectée, la couleur trouvée pour ce pixel est copiée sur tous les autres pixels appartenant au groupe.

La troisième étape de l'approche fovéale consiste à subdiviser à nouveau les groupes de pixels de chaque région. Comme le montre la figure 5.12 c), c'est à cette étape que la région centrale atteint sa résolution maximale. La région intermédiaire est maintenant formée de groupes de 2x2 pixels et la région extérieure est composée de groupes de 4x4 pixels. La quatrième étape (figure 5.12d)) termine l'affichage de la zone intermédiaire et ce n'est qu'à la cinquième étape que le pourtour de l'image est affiché à sa résolution maximale.

Les temps de rendu présentés à la figure 5.13 montrent qu'une image de qualité pleine résolution de la surface en cours de construction peut être obtenue en seulement quelques secondes. Il suffit à l'utilisateur d'interrompre la saisie de données quelques secondes pour que le système soit en mesure de rendre une vue haute résolution de la surface. Ainsi, l'utilisateur peut rapidement apprécier la qualité de la reconstruction de la surface et corriger la situation s'il découvre un problème quelconque. La figure 5.14 montre les images affichées à l'utilisateur pour chaque étape de l'approche fovéale. Notez que l'image fournie suite à la troisième phase est de très bonne qualité et est obtenue en environ 1,5 seconde.

Étant donné que le temps de rendu d'une image complète est de l'ordre de quelques secondes, il est important de pouvoir interrompre la génération d'une image dès qu'une interaction avec l'interface usager est requise. Cela est nécessaire pour garantir la réactivité du système. Lorsque l'utilisateur désire modifier l'angle d'observation du modèle, via la souris, le rendu en cours est interrompu et il n'y a que la première phase de l'approche fovéale qui est exécutée tant que le point de vue varie. Dès que l'utilisateur cesse de modifier l'angle d'observation, les phases subséquentes sont calculées et l'image haute résolution de la surface est affichée.

Une stratégie similaire est utilisée lors de l'acquisition des données pour éviter que la majeure partie des ressources processeur soit consacrée à l'affichage de la surface au lieu de la reconstruction du champ vectoriel. Cette stratégie consiste à ne mettre l'image à jour qu'une seule fois par seconde tout en ne permettant que la première phase de l'approche fovéale. Évidemment, dès que l'ajout de nouvelles données au modèle cesse, les phases subséquentes du rendu fovéal sont calculées pour afficher une image haute résolution dans le but de permettre à l'utilisateur d'apprécier la qualité de la surface reconstruite.

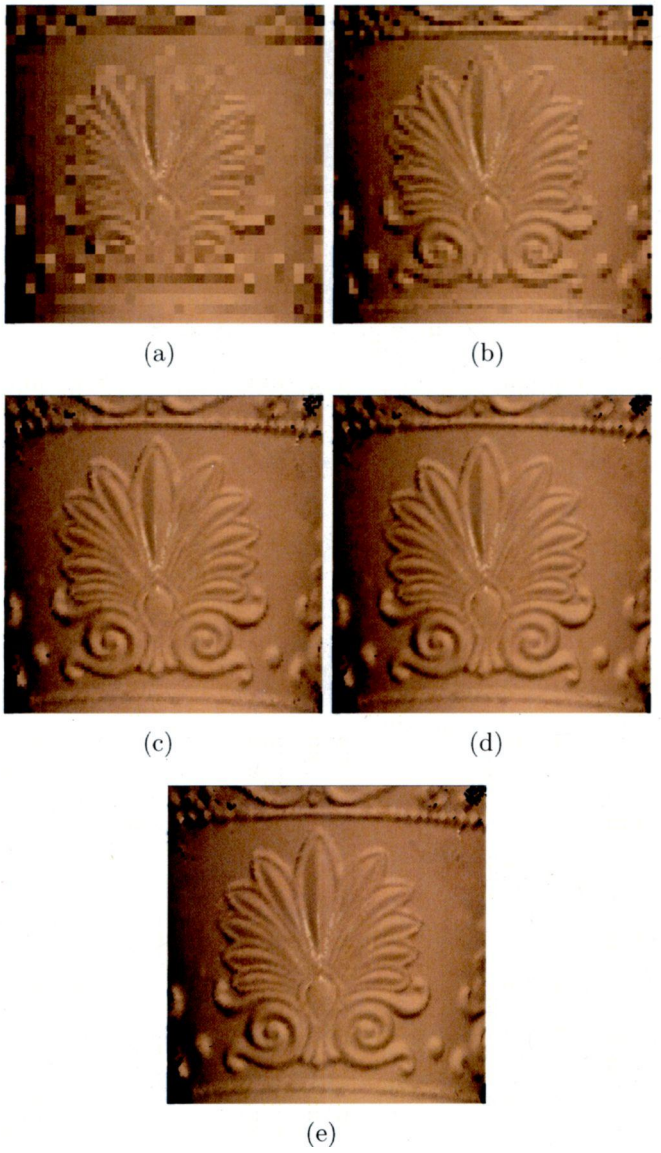


FIG. 5.14 – Aperçus de chacune des phases (1 à 5) de l'approche fovéale pour le rendu d'un objet réel, soit le vase utilisé au chapitre précédent.

5.4 Conclusion

Ce chapitre a présenté le fonctionnement de la dernière étape de la boucle de modélisation interactive, soit la visualisation de la surface en cours de reconstruction. Le module de visualisation doit fournir un rendu de qualité pour permettre à l'utilisateur de vérifier la couverture complète de l'objet, mais également d'apprécier la qualité de la reconstruction. Pour obtenir une telle qualité, l'approche choisie repose sur le lancer de rayons qui affiche directement les données contenues dans le champ vectoriel.

Le principe de fonctionnement de l'approche par lancer de rayons est simple. Pour chaque pixel du plan image, un rayon est lancé au travers du volume jusqu'à ce que la surface soit rencontrée. Plusieurs techniques de détection de la présence d'une surface ont été étudiées. Celle retenue se nomme *première entrée* et offre la plus grande qualité de rendu tout en évitant des artéfacts de visualisation tel que des faux tunnels ou un effet de crenélage sur le contour des modèles. En plus de détecter explicitement la présence d'une surface à l'intérieur des limites d'un *voxel,* cette technique vérifie pour chaque frontière entre deux *voxels* traversés par le rayon si un passage par zéro de la norme du champ vectoriel est présent. Si c'est le cas, le système conclut qu'il y a présence d'une surface à cet endroit.

Malgré le fait que le module de visualisation par lancer de rayons soit de complexité constante pour une taille d'image donnée et une profondeur fixe de l'*octree*, il n'en reste pas moins que la visualisation est l'étape la plus lourde de la boucle de modélisation interactive. La réactivité du système repose donc sur une approche d'affichage fovéale concentrant les premiers rayons lancés dans la zone centrale de l'image, zone contenant habituellement les détails d'intérêt que l'utilisateur veut voir. Il est évidemment possible d'obtenir une image pleine résolution de haute qualité en attendant quelques secondes.

Chapitre 6

Conclusion

La contribution principale de ces travaux est la mise au point d'un système de modélisation interactive 3D multirésolution permettant l'utilisation de la compression en temps réel. La compression en temps réel permet d'optimiser les ressources mémoire disponibles durant la séance d'acquisition. Cette utilisation judicieuse de la mémoire permet dorénavant de modéliser des objets de plus grande taille ou à plus haute résolution. Il a en effet été montré qu'il est dorénavant possible de reconstruire localement des régions détaillées d'un objet à des niveaux de résolution ne pouvant pas être atteint par des approches utilisant un niveau de résolution unique.

Le système présenté repose sur une nouvelle représentation multirésolution de surface 3D inspirée des travaux de Tubic[30]. Ce dernier a montré que pour résoudre le problème de la modélisation interactive, il ne faut pas seulement développer des algorithmes performants, mais plutôt utiliser une représentation de surface adaptée au problème. La représentation de surface qu'il a proposée correspond à une représentation implicite encodant un champ vectoriel. En chaque point d'une grille régulière, le champ vectoriel encode la direction et la distance du point le plus près sur la surface. L'avantage principal du champ vectoriel pour la modélisation interactive est la possibilité, lors de l'ajout d'un nouveau point 3D au modèle, d'accomplir chacune des étapes de la boucle de modélisation interactive en temps constant.

La représentation multirésolution proposée dans ces travaux correspond également à un champ vectoriel. Le point important qui la différencie de celle utilisée par Tubic *et al.* est la possibilité de reconstruire localement la surface à différents niveaux de résolution. La cohabitation des différents niveaux de résolution à l'intérieur de la même représentation a soulevé un problème important : la *cohérence* de la surface reconstruite. Nous avons défini une surface cohérente comme une surface pouvant être interprétée

correctement, en évitant les défauts dus à la multirésolution. L'interprétation de la surface peut prendre différentes formes comme la visualisation ou la génération d'une triangulation.

Après le développement de la représentation multirésolution, nous avons proposé un algorithme permettant de reconstruire localement la surface à un niveau de résolution optimal. Ici, le niveau de résolution optimal est celui permettant de bien capter les détails de la géométrie de l'objet tout en utilisant un minimum de mémoire. Les régions contenant de faibles courbures doivent être reconstruites à un niveau de résolution grossier et les régions contenant des détails à un niveau de résolution plus fin. Nous avons vu que la détermination du niveau de résolution optimal se fait en deux étapes.

La première étape correspond à la *reconstruction progressive* de la surface de l'objet en partant du niveau le plus grossier jusqu'au niveau le plus fin. L'intervalle des niveaux de résolution utilisé pour la reconstruction de la surface est fixé par l'utilisateur, au début de la séance d'acquisition. C'est la densité de points recueillie par l'utilisateur qui indique au système le niveau de résolution à atteindre. Un balayage rapide d'une région à faible courbure limitera la reconstruction de la surface à des niveaux de résolution grossiers tandis qu'un balayage lent et persistent permettra la reconstruction de la surface à un niveau de résolution élevé. Cela est dû au fait que la reconstruction de la surface à un certain niveau n'est permise que si le niveau précédent a été échantillonné suffisamment. Plus spécifiquement, nous avons vu que déterminer si une région a été échantillonnée suffisamment fait intervenir la notion de *stabilité*. Une région de surface *stable* est une région où la densité de points recueillie est suffisante pour éviter que la représentation encodée soit influencée de façon majeure lors de l'ajout de points supplémentaires.

La deuxième étape permettant de déterminer le niveau de résolution optimal auquel la surface doit être reconstruite correspond à la *compression en temps réel*. Cette étape permet de compresser les régions de la surface contenant peu de détail qui ont été reconstruites à un niveau de résolution trop élevé. Cette situation arrive fréquemment dans le pourtour des régions balayées intensément à l'aide d'un capteur 3D tenu en main. Le princpe derrière la *compression temps réel* est de comparer les représentations de la surface pour deux niveaux adjacents en commençant par le niveau le plus fin. Si les deux représentations représentent la même surface en fonction de la *condition d'ajustement*, la surface encodée au niveau fin n'est pas nécessaire et peut être éliminée. Cette étape permet donc de libérer de la mémoire durant la séance d'acquisition, toujours dans le but de modéliser des objets de plus grande taille ou à plus haute résolution.

Nous avons validé le système de modélisation interactive multirésolution avec des

jeux de données réels et synthétiques. Ces expériences ont montré que pour optimiser au maximum les ressources disponibles, il faut utiliser une approche d'acquisition par parties. Cette approche consiste à modéliser l'objet en plusieurs étapes, chacune se concentrant sur une partie de la surface. Avant de passer à la modélisation de la partie suivante, il faut recueillir suffisamment de points pour permettre l'activation de la *compression en temps réel* et ainsi libérer progressivement de la mémoire. Cette approche a montré une diminution de la consommation de mémoire de plus de 50% comparée à une approche tentant de modéliser l'objet de façon uniforme.

De plus, nous avons montré qu'il est possible de reconstruire des détails fins d'un objet à un niveau de résolution très élevé tout en permettant la reconstruction du contexte et l'obtention d'un modèle 3D complet de l'objet en question. Lors de la simulation, les régions contenant beaucoup de détails ont été reconstruites à un niveau de résolution quatre fois plus fin que celui pouvant être atteint par les approches utilisant une représentation à un niveau de résolution unique. Une des difficultés rencontrée lors de cette simulation a été de générer une densité de points suffisante pour permettre la reconstruction de la surface au niveau de résolution maximum. De plus, les données utilisées doivent avoir un niveau de bruit inférieur à la distance séparant deux noeuds adjacents du niveau le plus fin.

La dernière partie de ces travaux a été consacrée au module de visualisation de la surface en cours de reconstruction. La fonction première du module de visualisation est de permettre d'évaluer si les données brutes recouvrent la totalité de l'objet. Une autre fonction très importante de la visualisation est de valider la qualité de la reconstruction. Pour ce faire, le module de visualisation doit offrir des rendus de qualité et être performant.

Nous avons vu que la visualisation d'un champ vectoriel peut se faire à l'aide de deux techniques différentes. Premièrement, les techniques basées sur la *projection de pastilles* consistent à projeter une pastille sur le plan image pour chaque position du champ vectoriel contenant une représentation d'une section de surface valide. Cette approche n'est pas très bien adaptée à l'affichage d'une surface en cours de reconstruction, car elle nécessite un traitement suite à l'acquisition des données pour fournir un rendu de qualité. Dans le contexte de la modélisation interactive, ce genre de traitement n'est pas envisageable.

La technique utilisée repose plutôt sur une approche par *lancer de rayons*. Cette approche consiste à lancer un rayon au travers du volume pour chaque pixel du plan image. Lorsque le rayon rencontre la surface, la couleur du pixel est déterminée en fonction de l'angle qu'il fait avec la normale à la surface. Implanter une approche par

lancer de rayons peut se faire de façon très efficace en utilisant une technique exploitant la représentation paramètrique des rayons.

Malgré cette implantation efficace, le rendu par lancer de rayons demande beaucoup de ressources. C'est pour cette raison qu'une approche fovéale a été développée. Cette approche concentre les premiers rayons lancés dans le centre de l'image, région contenant généralement les détails d'intérêt que l'utilisateur veut voir. Avec cette technique, il est possible d'offrir un aperçu de la surface plusieurs fois par seconde. Quelques secondes suffisent pour obtenir un rendu de qualité permettant de déterminer la qualité de la reconstruction de la surface.

Travaux futurs

Actuellement, le système ne permet pas de traiter plus de 2000 points par seconde. Pour rendre le système plus réactif, il faudrait accélérer le tout d'un facteur 10. L'étape consommant beaucoup de ressources processeur est la *compression en temps réel,* plus précisément l'évaluation de la *condition d'ajustement* dans le voisinage d'une section de surface. Le champ vectoriel est présentement encodé dans un *octree* et cette structure de données ne permet pas de facilement visiter les noeuds voisins qui appartiennent à une autre branche. Il faudrait donc modifier l'*octree* pour permettre une exploration efficace de ces voisins.

Un autre aspect du système proposé pouvant être amélioré concerne la manière dont chaque niveau de résolution est encodé. Présentement chacun des niveaux de résolution est indépendant de ses parents. Pour chaque région de l'objet, tous les niveaux de résolution compris entre le niveau grossier et le niveau le plus fin disponible à cet endroit contiennent une représentation valide de la surface. Les données d'un niveau fin ne reposent pas sur les données encodées à un niveau plus grossier. Il serait intéressant de développer une représentation s'appuyant sur les niveaux grossiers pour encoder les niveaux de résolution plus fins, un peu à la manière des ondelettes. Pour un niveau fin, l'idée serait d'encoder la différence entre sa représentation et celle du niveau parent. Cependant, mettre en oeuvre cette technique n'est pas simple à cause du contexte de la modélisation interactive : l'utilisateur peut toujours ajouter des points au modèle. Il faut donc une approche qui est en mesure de gérer un flot continu de données.

Finalement, il serait intéressant de développer un module permettant d'extraire une triangulation à partir d'un champ vectoriel multirésolution. L'extraction d'une triangulation est nécessaire pour permettre l'utilisation des modèles obtenus dans diverses

applications. Les exemples typiques sont l'utilisation des modèles dans des moteurs 3D conventionnels ainsi que dans des logiciels de modélisation basés sur des représentations par triangles.

Annexe A

Implantation de la traversée paramétrique de l'*octree*

Trouver la liste des octants enfants traversés par un rayon peut être de beaucoup simplifié par un bon choix d'identification des octants enfants. Comme le montre la figure A.1, l'index des octants enfants, codés sur trois bits, décrit directement leur position dans le volume du parent. Le bit de poids faible implique une translation selon l'axe des X, le second, selon l'axe des Y et le dernier selon l'axe des Z. Pour trouver l'octant dans lequel le rayon pénètre en premier, il suffit de comparer la position 3D, selon chaque axe, du point d'entrée avec la position 3D du centre du noeud parent. Si le point d'entrée est situé à une valeur supérieure selon l'axe des X que le centre du *voxel*, le bit de poids faible est mis à un. Même chose pour l'axe des Y et le second bit ainsi que l'axe des Z et le dernier bit. L'index ainsi obtenu correspond donc à l'index de l'octant dans lequel le rayon pénètre initialement.

Une fois le premier octant traversé par le rayon identifié, il ne reste plus qu'à trouver la liste des octants par lesquels le rayon passe avant de quitter le noeud parent.

Pour passer d'un octant enfant à un autre, il suffit de manipuler les bits de son index de la manière suivante. Si le plan de sortie du rayon est XY, cela signifie que le prochain octant enfant traversé implique un déplacement selon l'axe des Z. Donc, il suffit de changer l'état du 3e bit de l'index de l'octant courant en appliquant l'opération XOR 4 (XOR 100 en binaire). Dans le cas où le plan de sortie est XZ, il suffit d'appliquer l'opération XOR 2 (XOR 010 en binaire) à l'index courant. Finalement, si le plan de sortie est YZ, appliquer l'opération XOR 1 (XOR 001 en binaire) à l'index courant fournira l'index du prochain octant traversé. La traversée des octants enfants se termine lorsque le rayon a rejoint le paramètre λ_{sortie}. À ce moment, le rayon quitte le noeud

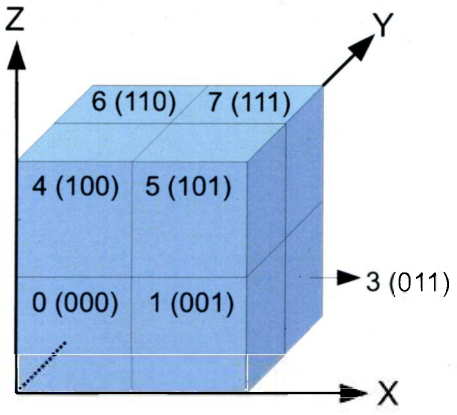


FIG. A.1 – Identification des octants d'un noeud parent par leur position selon les trois axes du système de coordonnées. L'octant le plus près de l'origine prend l'index 0. Celui adjacent, sur l'axe des X, correspond à l'index 1 (001). Ceux adjacents sur les axes des Y et des Z prennent respectivement l'index 2 (010) et l'index 4 (100).

racine.

L'exemple présenté à la section 5.2.2 a montré comment il est possible de déterminer les points d'entrée et de sortie du rayon pour chaque octant enfant lors de la traversée du noeud racine. Dans le cas d'un champ vectoriel multirésolution, il faut que l'algorithme soit en mesure de déterminer progressivement tous les points d'entrée et de sortie du rayon et ce, pour tous les niveaux disponibles de l'*octree*. Pour ce faire, il suffit d'appliquer l'algorithme proposé récursivement à chaque enfant appartenant au noeud racine tant que le niveau le plus fin localement n'est pas atteint. Ainsi, de manière récursive, tous les noeuds traversés par le rayon seront identifiés. La progression du rayon arrêtera lorsque la surface aura été atteinte ou que le rayon aura quitté le volume de la racine.

L'appel récursif de cet algorithme sur chaque octant enfant nécessite de déterminer quels paramètres, sur le rayon, correspondent aux paramètres limites de l'octant enfant, c'est-à-dire $\lambda_{min_x}, \lambda_{min_y}, \lambda_{min_z}$ et $\lambda_{max_x}, \lambda_{max_y}, \lambda_{max_z}$. Les paramètres limites pour un octant dépendent de l'index de l'octant ainsi que de la direction du rayon lancé. Au lieu de calculer les paramètres intermédiaires en utilisant l'équation $\frac{\lambda_{min}+\lambda_{max}}{2}$, nous pouvons écrire :

$$\lambda' = \lambda_{min} + \frac{(\lambda_{max} - \lambda_{min})}{2} = \lambda_{min} + \frac{\delta}{2} \tag{A.1}$$

Donc, les paramètres limites des enfants peuvent s'écrire à partir des paramètres limites du noeud parent et de la variation du paramètre (δ) pour chacun des axes. Prenons l'exemple de l'octant ayant un index de 0 (000). Ses paramètres limites sont $(\lambda_{min_x}, \lambda_{min_y}, \lambda_{min_z})$ et $(\lambda_{min_x} + \frac{\delta_x}{2}, \lambda_{min_y} + \frac{\delta_y}{2}, \lambda_{min_z} + \frac{\delta_z}{2})$. Pour le noeud ayant l'index 1 (001), les paramètres limites sont $(\lambda_{min_x} + \frac{\delta_x}{2}, \lambda_{min_y}, \lambda_{min_z})$ et $(\lambda_{min_x} + 2\frac{\delta_x}{2}, \lambda_{min_y} + \frac{\delta_y}{2}, \lambda_{min_z} + \frac{\delta_z}{2})$. Notez ici que $\lambda_{min_x} + 2\frac{\delta_x}{2}$ correspond en fait à λ_{max_x}. Avec cette approche, il est maintenant possible de calculer les paramètres limites en décomposant chacun des bits de l'index i de l'octant de la façon suivante :

$$\gamma_x = i\&1 \tag{A.2}$$

$$\gamma_y = (i/2)\&1 \tag{A.3}$$

$$\gamma_z = (i/4)\&1 \tag{A.4}$$

où & correspond à l'opérateur logique ET et l'opération de division est appliquée sur des entiers. Les paramètres limites minimums deviennent donc $(\lambda_{min_x} + \gamma_x \frac{\delta_x}{2}, \lambda_{min_y} + \gamma_y \frac{\delta_y}{2}, \lambda_{min_z} + \gamma_z \frac{\delta_z}{2})$ et les paramètres limites maximums $(\lambda_{min_x} + (\gamma_x + 1)\frac{\delta_x}{2}, \lambda_{min_y} + (\gamma_y + 1)\frac{\delta_y}{2}, \lambda_{min_z} + (\gamma_z + 1)\frac{\delta_z}{2})$. Cette façon de faire évite l'utilisation d'une série d'opérateurs IF imbriqués et simplifie grandement le code.

Cette technique fonctionne uniquement si les composantes du vecteur pointant dans la direction du rayon sont de valeurs positives. Si une des composantes est négative, il faut ajuster les valeurs de γ pour tenir compte du fait que le rayon entre dans l'*octree* par le plan T et non pas par le plan passant par l'origine. Il suffit simplement d'inverser la valeur de γ pour les axes ayant des composantes négatives. Cela revient à dire que pour un rayon ayant une composante négative selon l'axe des X, le bit de poids faible de son index d'entrée sera inversé. Le même principe s'applique pour l'axe des Y et le second bit de l'index ainsi que pour l'axe des Z et le bit de poids fort de l'index.

Bibliographie

[1] Besl, P. and McKay, N., A Method for Registration of 3-D Shapes, in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2), pp.239-256, February, 1992.

[2] Binotto, A.P.D., Comba J.L.D. and Freitas, C.M.D., Real-Time Volume Rendering of Time-Varying Data Using a Fragment-Shader Compression Approach, in *Proceedings of the 2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, 10 pages, October 20-21, 2003.

[3] Boada, I. and Navazo, I., Multiresolution Isosurface Fitting on a Surface Octree, in *Proceedings of the Vision Modeling and Visualization Conference 2001*, Stuttgart, Germany, pp.317-324, November 21-23, 2001.

[4] Bodenmueller, T. and Hirzinger, G., Online Surface Reconstruction from Unorganized 3D-Points for the DLR Handguided Scanner System, in *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'04)*, Thessaloniki, Greece, pp.175-182, September 6-9, 2004.

[5] Chao, C., Stamos, I., Semi-Automatic Range to Range Registration : a Feature-Based Method, in *Proceedings of the Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM),* Ottawa, Canada, pp.254-261, June 13-16, 2005.

[6] Curless, B. and Levoy, M., A Volumetric Method for Building Complex Models from Range Image, in *SIGGRAPH '96 Proceedings*, New Orleans, USA, pp.303-312, August 4-9, 1996.

[7] Deschênes, J.D., Hébert, P., Lambert, P., Ouellet, J.N. and Tubic, D., Multiresolution Interactive Modeling with Efficient Visualization, in *Proceedings of the Fifth International Conference on 3-D Digitial Imaging and Modeling (3DIM)*, Ottawa, Canada, pp.39-46, June 13-16, 2005.

[8] Deschênes, J.D., Lambert, P., Hébert, P., Interactive Modeling with Automatic Online Compression, in *Proceedings of the Third International Symposium on 3D Data Processing, Visualization and Transmission(3DPVT)*, Chapel Hill, USA, pp.766-773, June 14-16, 2006.

[9] Fiocco, M., Bostrom, G., Goncalves, J.G.M. and Sequeira, V., Multisensor fusion for Volumetric Reconstruction of Large Outdoor Areas, in *Proceedings of the Fifth International Conference on 3D Digital Imaging and Modeling (3DIM)*, Ottawa, Canada, pp.13-17, June 13-16, 2005.

[10] Ghazanfarpour, G. and Hasenfratz. J. M., A Beam Tracing with Precise Antialiasing for Polyhedral Scenes, in *Computer and Graphics*, 22(1), pp.103-115, February, 1998.

[11] Hébert, P., A Self-Referenced Hand-Held Range Sensor, in *Proceedings of the Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, Quebec, Canada, pp.5-11, May 28-June 1, 2001.

[12] Heckbert, P.S. and Hanrahan, P., Beam Tracing Polygonal Objects, *ACM SIGGRAPH Computer Graphics*, 18(3), pp.119-127, July, 1984.

[13] Hilton, A., and Illingworth, J., Geometric Fusion for a Hand- Held 3D Sensor, in *Machine Vision and Applications*, 12, pp.44-51, July, 2000.

[14] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W., Surface Reconstruction from Unorganized Points, in *SIGGRAPH '92 Proceedings*, Chicago, USA, pp.71-78, July 26-31, 1992.

[15] Ju, T., Losasso, F., Schaefer, S. and Warren, J., Dual Contouring of Hermite Data, in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, San Antonio, USA, pp.339-346, July 21-26, 2002.

[16] Kazhdan, M., Bolitho, M. and Hoppe, H., Poisson Surface Reconstruction, in *Symposium on Geometry Processing,* Sardinia, Italy, pp.61-70, June 26-28, 2006.

[17] Kruger J. and Westermann R., Acceleration Techniques for GPU-based Volume Rendering, in *Proceedings of IEEE Visualization '03*, Seattle, USA, pp.287-292, October 24, 2003.

[18] Lee, C.H. and Park, K.H., Fast Volume Rendering Using Adaptive Block Subdivision, in *Pacific Graphics*, Seoul, Korea, pp.148-158, October 13-16, 1997.

[19] Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton. M., Anderson, S., Davis, J., Ginsberg, J., Shade, J. and Fulk, D., The Digital Michelangelo Project : 3D Scanning of Large Statues, in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, New Orleans, USA, pp.131-144, July 23-28, 2000.

[20] Levoy M., Efficient Ray Tracing of Volume Data, in *ACM Transactions on Graphics (TOG)*, 9(3), pp.245-261, July, 1990.

[21] Lorensen W.E. and Cline, H. E., Marching cubes : A High Resolution 3D Surface Construction Algorithm, in *Computer Graphics SIGGRAPH '87 Proceedings*, Anaheim, USA, pp.163-169, July 27-31, 1987.

[22] Phong, B.T., Illumination for Computer-Generated Images, Department of Computer Science, Utah University, 1975.

[23] Revelles, J., Ureña, C. and Lastra, M., An Efficient Parametric Algorithm for Octree Traversal, in *Proceedings of the 8th International Conference on Computer Graphics and Visualization 2000*, Plzen, Czech Republic, pp.212–219, February 7-10, 2000

[24] Rössl, C., Zeilfelder, F., Nürnberger, G. and Seidel, H.-P., Reconstruction of Volume Data with Quadratic Super Splines, in *IEEE Transactions on Visualization and Computer Graphics*, 10(4), pp.397-409, July-August, 2004.

[25] Rusinkiewicz, R., Hall-Holt, O. and Levoy, M., Real-Time 3D Model Acquisition, in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, San Antonio, USA, pp.438-446, July 21-26, 2002.

[26] Rusinkiewicz, S. and Levoy, M., QSplat : A Multiresolution Point Rendering System for Large Meshes, *in Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, New Orleans, USA, pp.343-352, July 23-28, 2000.

[27] Schaefer, S., and Warren, J., Dual Contouring : "The Secret Sauce". Technical report, Rice University, 2002.

[28] Szeliski, R. and Lavallée, S., Matching 3-D Anatomical Surfaces with Non-Rigid Deformations using Octree-Splines, in *International Journal of Computer Vision*, Springer, 18(2), pp.171-186, 1996.

[29] Thevenaz, P. and Unser, M., Precision Isosurface Rendering of 3-D Image Data, in *IEEE Transactions on Image Processing*, 12(7), pp.764-775, July, 2003.

[30] Tubic, D., On Surface Representation in 3D Modelling A Framework for Interactive 3D Real-time Modelling. *Thèse de doctorat*, Université Laval, 2006.

[31] Tubic, D., Hébert, P., Deschênes, J.D. and Laurendeau, D., A Unified Representation for Interactive 3D Modeling, in *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT)*, Thessaloniki, Greece, pp.175-182, September 6-9, 2004.

[32] Tubic, D., Hébert, P. and Laurendeau, D., A Volumetric Approach for the Registration and Integration of Range Images : Towards Interactive Modeling Systems, in *Proceedings of the 16th International Conference on Pattern Recognition (ICPR)*, Quebec, Canada, 3(3), pp.283-287, August 11-15, 2002.

[33] Wald, I. and Slusallek, P., State of the Art in Interactive Ray Tracing, in *Proceedings of EUROGRAPHICS 2001*, Manchester, United Kingdom, pp.21-42, September 4-7, 2001.

[34] Zwicker, M., Pfister, H., Van Baar, J. and Gross, M., EWA Splatting, in *IEEE Transactions on Visualization and Computer Graphics*, 8(3), pp.223-238, July-September 2002.

[35] Creaform, http ://www.creaform3d.com/francais/01.3_news.html

[36] MCG3D, http ://www.mcg3d.com/accueil.php3 ?id_article=2

[37] The Matrix Trilogy, http ://whatisthematrix.warnerbros.com/

[38] XYZ-RGB Imaging Services, http ://www.xyzrgb.com/

[39] XYZ-RGB, A PlayStation 3 Game That Uses 4K Production Techniques Seen In The Matrix & King Kong Films, http ://www.xyzrgb.com/html/ps3_f1.pdf

[40] Steel University, http ://www.steeluniversity.org/

[41] Site de Jean-François Ortiz, http ://jf.ortiz.free.fr/basreliefes.htm