N° d'ordre: 639

UNIVERSITÉ DE STRASBOURG

École Doctorale Mathématiques, Sciences de l'Information et de l'Ingénieur

Université de Strasbourg

THÈSE

présentée pour obtenir le grade de

Docteur de l'Université de Strasbourg
Discipline : Informatique

par

Lucas Ammann

Visualisation temps réel de données à deux dimensions et demie

Soutenue publiquement le 17 septembre 2010

Membres du jury

Mme. Dominique Bechmann ..... *Présidente du Jury*
Professeure à l'Université de Strasbourg

M. Jean-Michel Dischler ..... *Directeur de Thèse*
Professeur à l'Université de Strasbourg

M. Éric Galin ..... *Rapporteur*
Professeur à l'Université Lumière, Lyon 2

M. Bruno Lévy ..... *Rapporteur*
Directeur de Recherche à l'INRIA Nancy - Grand Est

M. Tamy Boubekeur ..... *Examinateur*
Maître de Conférences à Telecom ParisTech, Paris

Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection **UMR 7005 CNRS-UdS**

Remerciements

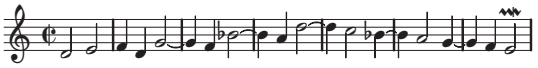
Je tiens avant toutes choses à remercier un certain nombre de personnes pour leur soutient, leur implication et leur aide durant ces années de thèse.

Je souhaite ainsi remercier M. Jean-Michel Dischler pour m'avoir accepter pour cette thèse. Je remercie également les membres de mon jury, Mme Dominique Bechmann, MM. Bruno Lévy, Éric Galin et Tamy Boubekeur pour avoir bien voulu s'intéresser à mes travaux.

Je voudrais adresser des remerciements tous particuliers à MM. Frédéric Larue et Olivier Génevaux avec qui j'ai eu la chance (et le privilège) de pouvoir travailler et qui se sont toujours montrés d'un grand secours tout au long de cette thèse qui aurait sûrement pris une autre voie sans eux. Merci à eux et à leur conseils toujours éclairés et pertinents.

Merci aux collègues pour l'ambiance générale de travail (studieuse mais pas trop quand même) et notamment à Simon E.B., Thomas, Manu, Guillaume ainsi que Nicolas, Julien et Pascal. Merci à eux de m'avoir supporté (et ma musique) pendant ces années. Merci plus généralement aux différents membres de l'équipe IGG.

Et pour terminer, je ne voudrais surtout pas oublier de remercier ma famille et en particulier mes parents Marc et Martine ainsi que ma sœur Marie-Alix pour m'avoir toujours soutenu et aidé pendant ces longues années.



J.S. Bach, thème de la fugue "Dorienne" BWV 538

Table des matières

Chapitre 1

Introduction

À l'heure actuelle, l'imagerie en trois dimensions est de plus en plus présente dans notre quotidien. L'industrie du divertissement, qui inclut les domaines d'activités liés aux développements des jeux vidéos, ou à la création de films ayant recours à des technologies informatiques, en est un exemple fort. Cependant, le domaine de l'informatique graphique, et plus particulièrement celui de la visualisation de données et du rendu de celles-ci, représente un secteur très vaste d'applications pouvant couvrir des domaines extrêmement variés qui s'étendent de l'exploration de données, provenant de sources diverses (résultats de simulations physiques, données topographiques, ...), à des applications plus culturelles, comme l'étude du patrimoine disparu ou existant (œuvres d'arts, études de pièces ou de lieux archéologiques, ...) ou de divertissements (jeux vidéos, film d'animations, ...). Ces différents domaines d'applications ont des besoins de plus en plus importants. En effet, ils nécessitent des outils toujours plus développés afin de visualiser les différentes données qu'ils ciblent. Cette complexité découle, en partie, des masses données sans cesse plus importantes et complexes qui sont générées et manipulées.

Le fort accroissement de ces besoins, est, en partie, le résultat des développements très actifs des dispositifs matériels d'acquisition, de traitements et de visualisations de ces différentes données. Durant ces dernières années, les possibilités offertes par les matériels graphiques ont été décuplées, aussi bien en termes de puissance brute de calculs que de flexibilité, ouvrant la voie à de nouvelles applications toujours plus sophistiquées, permettant de repousser les limites des techniques de visualisation.

Cependant, afin de fournir des résultats visuels qui soient en adéquation avec les attentes des différents utilisateurs, il est nécessaire de traiter les données en fonction de leurs propriétés. Ces dernières peuvent varier suivant leur provenance et de leur usage final. Ainsi, afin d'adapter au mieux les outils de

visualisation, il existe un grand nombre de type de représentations des données, ayant chacune leurs spécificités et leurs domaines d'applications. On peut, à titre d'exemple, citer les données volumiques utilisées dans le domaine médical, la représentation de surfaces en trois dimensions via des maillages pour les jeux vidéos, ou encore les données à deux dimensions et demie permettant de représenter des données topographiques.

1 Problématique générale

Dans le cadre des travaux menés durant cette thèse, nous nous sommes attachés à la visualisation d'un type de données très particulier et très largement exploités dans certains domaines, comme la visualisation de données scientifiques ou les applications liées à la géomatique. Il s'agit des données représentant des objets qui peuvent être exprimés avec seulement deux dimensions et demie. De plus, ce format de données peut également être généré par certains dispositifs de numérisation à trois dimensions.

1.1 Contexte

Les données à deux dimensions et demie permettent de représenter des objets assez simples mais dont l'utilisation est très fréquente et dont les domaines d'applications sont extrêmement variés. Ces données autorisent la modélisation d'un ensemble de sommets, définis dans un espace à trois dimensions, en ne spécifiant que leur distance par rapport à un plan de base. Cela permet de représenter une information complexe, définissant une surface dans un espace à trois dimensions, en ne stockant qu'une valeur qui représente une distance par sommet, et un plan de base, commun à tous les points formant l'objet. Cette forme de représentation est, au final, assez compacte et peut être utilisée sur des masses de données très importantes.

Ce type de données est principalement employé pour la représentation de reliefs formant des terrains. En effet, elles sont très fréquemment utilisées pour modéliser des terrains. Elles peuvent, par exemple, être issues de la numérisation de reliefs effectuée par des satellites, mesurant l'altitude de la surface de certaines régions de la Terre. En plus de ces données topographiques, elles peuvent également être générées par des processus de simulations physiques décrivant leurs résultats via des champs scalaires assimilables à des cartes de distances, définies en fonction d'un plan de base. Les données à deux dimensions et demie permettent donc de représenter une carte de distances depuis un plan de base. Cela en fait également une représentation tout à fait adaptée

à des dispositifs d'acquisition en trois dimensions qui se bornent généralement à mesurer la distance entre un point (le scanner) et l'objet numérisé. Ces dispositifs fournissent ainsi des images télémétriques où chaque pixel correspond à une information de distance et non pas de couleur. Par conséquent, certains dispositifs utilisent cette représentation qui, en plus, peut directement être exploitée pour visualiser les objets numérisés.

Le point commun entre toutes ces applications se résume par le fait que ces données peuvent représenter des masses de données non négligeables, où les cartes de distances peuvent dépasser plusieurs centaines de millions d'échantillons. Il est donc nécessaire de développer des techniques dédiées à ces données qui présentent, de plus, une structuration très particulière. En effet, elles sont caractérisées par un échantillonnage régulier, prenant la forme d'une simple grille à deux dimensions, où en chaque cellule est stockée une valeur représentant la distance au plan de base. Tout cela introduit des spécificités dont il faut tenir compte lors de leur visualisation : gestion de la masse de données, interpolation des échantillons pour reconstruire une surface, etc. À tout cela s'ajoute également des problèmes liés à l'échelle des données visualisées : représentent-t-elles des reliefs ayant une amplitude faible (à une échelle microscopique ou moindre, comme des micros reliefs d'une surface rugueuse) ou élevée (à une échelle macroscopique, comme une chaîne montagneuse par exemple).

1.2 Représentation des données : champs de hauteurs

D'un point de vue formel, les données à deux dimensions et demie sont des données qui représentent une fonction définie dans $\mathbb{R}^2$ et à valeurs dans $\mathbb{R}$. Elle associe à un couple de coordonnées une valeur utilisée généralement comme élévation [1] :

$$\begin{array}{rcl} h : \mathbb{R} \times \mathbb{R} & \rightarrow & \mathbb{R} \\ (x, y) & \mapsto & z \end{array}$$

Ce type de fonctions, généralement appelées champ de hauteurs [2], permet de représenter facilement une surface dans $\mathbb{R}^3$ définie par rapport à un plan de base, lui aussi défini dans $\mathbb{R}^3$. Ainsi, pour tout point P, de coordonnées (x_P, y_P, z_P) de ce plan de base, chaque point P' appartenant à la surface représentée par la fonction h est défini par :

$$P' = P + \overrightarrow{n} \cdot h(x_P, y_P)$$

où $\overrightarrow{n}$ représente la normale du plan de base, comme l'illustre la figure 1.1.

1. On peut retrouver également les termes d'altitude ou de hauteur pour désigner l'élévation.

2. Il existe tout un ensemble de synonymes, tels que carte de hauteurs, carte d'élévations, etc.

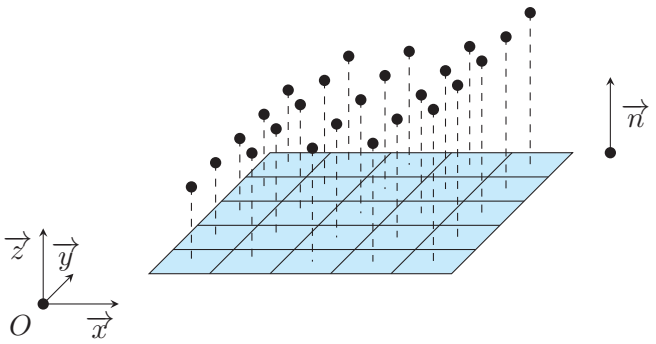


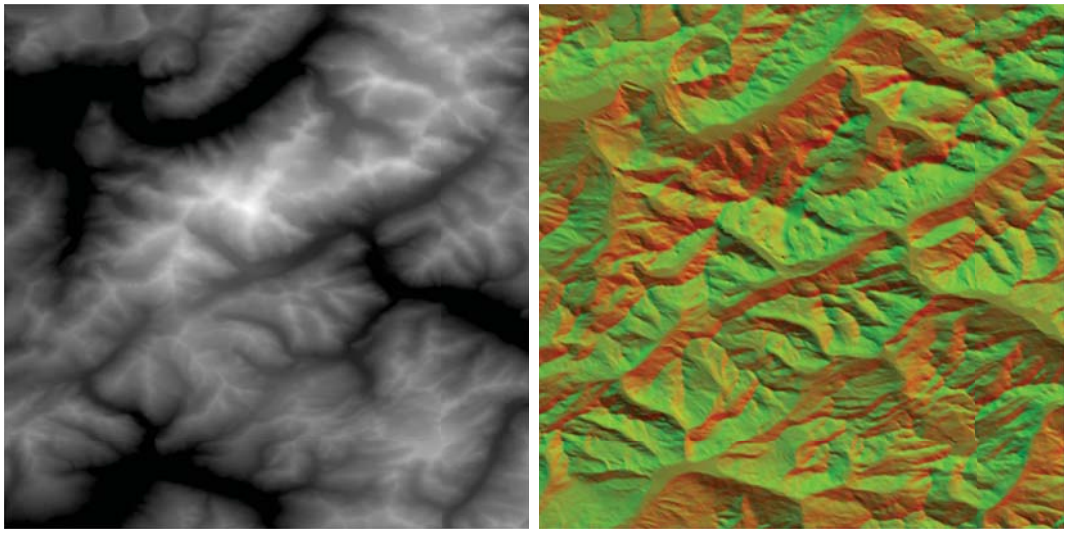
Figure 1.1 – Exemple de représentation d'une surface par un champ de hauteurs. La surface formée par l'ensemble de points en noir est définie par rapport à un plan de base, matérialisé en bleu, dont le vecteur normal est le vecteur $\vec{n}$.

La manipulation et le stockage d'une telle fonction d'un point de vue informatique ne peuvent s'effectuer que de façon discrète. Par conséquent, elles sont représentées par une simple grille à deux dimensions où chaque cellule contient une valeur d'élévation, comme sur l'exemple de la figure 1.1. D'un point de vue pratique et en vue de l'exploiter par le processeur graphique, cette grille est stockée en utilisant une texture à 2 dimensions. Un exemple d'une telle texture est donné par la figure 1.2a qui présente un champ de hauteurs représentant des données topographiques et plus spécialement la région du massif Mont-Blanc, situé dans les Alpes.

En plus des données constituant le champ de hauteurs, il peut être utile de stocker certaines informations supplémentaires en parallèle des élévations. Cela peut, par exemple, être le cas des vecteurs normaux de la surface. Ces données peuvent facilement être extraites du champ de hauteurs. Cette opération peut s'effectuer au moment de l'étape du rendu, ou durant une étape de pré-traitements. Dans ce cas, il est nécessaire de stocker ces valeurs. Là aussi, on utilise généralement une carte de normales qui est stockée via une texture à deux dimensions. L'utilisation de textures peut poser certaines contraintes quant à la représentation des données : il est généralement nécessaire de convertir les coordonnées des vecteurs normaux de l'intervalle $[-1; 1]$ vers l'intervalle $[0; 1]$ [3]. Un exemple de carte de normales, dont les coordonnées ont été normalisées dans l'intervalle $[0; 1]$, est donnée par la figure 1.2b. On peut noter que la carte de normales et la carte de hauteurs peuvent être fusionnées au sein d'une même texture à quatre composantes.

Comme dans un espace à trois dimensions, le champ de hauteurs est

3. Cette contrainte peut maintenant être levée par l'utilisation des textures de nombres flottants supportées désormais par la quasi-majorité des cartes graphiques actuelles.



(a) **(b)**

Figure 1.2 – Exemple de champs de hauteurs représentant des données topographiques (a) et sa carte de normales associée (b), où les couleurs représentent les coordonnées de ces vecteurs (n_x correspond à la composante rouge, n_y à la composante verte et n_z à la composante bleue), normalisée dans l'intervalle $[0; 1]$. Ces données représentent la région du massif du Mont Blanc.

dépendant d'un plan de base, les vecteurs normaux doivent aussi être exprimés en fonction de ce plan. On les exprime généralement dans l'espace tangent à celui-ci. Cet espace est formé par deux vecteurs définis dans le plan de base et le vecteur normal du plan.

2 Contexte matériel : quelques définitions

Afin de développer des méthodes de rendu et de visualisation performantes, aussi bien en terme de qualité visuelle que de vitesse, il faut désormais exploiter au maximum le matériel graphique et, en particulier, exploiter au maximum les capacités de programmation offertes par les processeurs graphiques actuels. Pour cela, une brève description de ce contexte matériel est nécessaire.

Chaîne de traitements graphiques La chaîne de traitements graphiques est restée très longtemps composée de différentes étapes dont le comportement ne pouvait être modifié que très peu, par un simple ajustement de seulement quelques paramètres. Désormais, certaines étapes de cette chaîne de traite-

ments sont devenues entièrement programmables au prix d'un certain nombre de contraintes liées au type même des processeurs graphiques (appelés également *GPU* pour *Graphics Processing Unit*). En effet, les processeurs graphiques sont des processeurs de type *SIMD* (*Single Instruction, Multiple Data*), appliquant, en parallèle, le même ensemble de traitements à un ensemble de données sans possibilité de dépendances entre les traitements des données. Concrètement dans le cas d'un GPU, cela se traduit par un ensemble de traitements identiques appliqués à tous les pixels d'une image, mais le traitement d'un pixel ne peut pas s'appuyer sur le résultat du pixel voisin.

Avec différents modèles de processeurs graphiques apparus vers 2007 [4], la chaîne de traitements graphiques se décompose suivant le schéma de la figure 1.3. Les données, permettant de débuter les traitements définis par cette architecture, sont les sommets de la géométrie à afficher. Ils sont, dans un premier temps, traités individuellement (au niveau du *vertex shader*) puis assemblés afin de former les différentes primitives géométriques (étapes d'assemblage de primitives et du *geometry shader*). Les parties de la géométrie ne rentrant pas dans la pyramide de vision sont éliminées avant que les parties valides soient rastérisées pour générer les pixels (ou *fragments*) [5] qui composent l'image finale. Afin de calculer leur couleur, un dernier traitement est appliqué sur chaque fragments (au niveau du *fragment shader*). La chaîne se termine par différents tests sur les fragments (profondeur, transparence, ...) avant leur écriture dans l'espace mémoire associée à la zone de dessin de l'écran (*frame buffer*) utilisée pour l'affichage.

On peut noter qu'il existe des interactions fortes entre le processeur central (ou *CPU* pour *Central Processing Unit*), la mémoire centrale, la mémoire graphique et le processeur graphique. En effet, le processeur central envoie les commandes de dessin au GPU et gère les échanges mémoires, entre mémoire centrale et mémoire graphique. Ces derniers constituent d'ailleurs généralement un goulot d'étranglements qui peut pénaliser les performances des méthodes exploitant ce type d'architecture.

Chaque étape programmable de chaîne de traitements graphiques est définie par un ensemble de programmes appelés *shaders* (ou programme de *shaders*). Parmi les différentes étapes programmables, on compte :

Vertex shader reçoit en entrée les sommets composant la géométrie à afficher. Chaque sommet peut être associé à différentes valeurs (couleurs, coordonnées de textures, normales à ces sommets, ...). Ce shader applique

4. Processeurs graphiques de type NVDIA G80.

5. À ce niveau de la chaîne de traitements graphique, on parle plutôt de fragments. En effet, un pixel est un fragment qui fait partie de l'image finale : un fragment n'est pas obligatoirement un pixel, et peut être évincé de l'image finale.

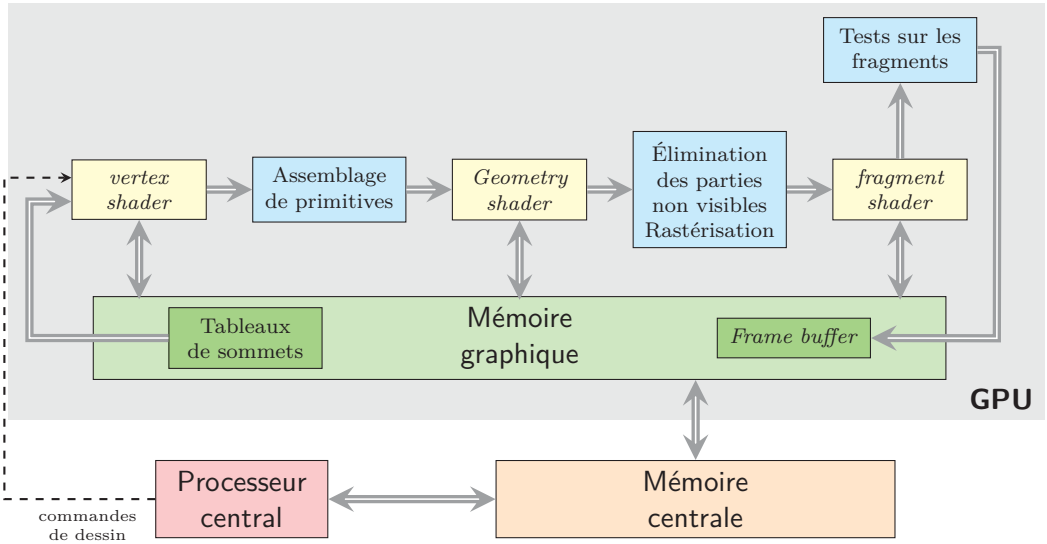


Figure 1.3 – Chaîne de traitements graphiques. Le point d'entrée de la chaîne se situe au niveau du *vertex shader* et est commandée par le processeur central. En jaune, les étapes programmables de cette chaîne.

un traitement sur chacun des sommets reçus.

Geometry shader situé après l'assemblage des sommets en différentes primitives (triangles, lignes, ...), il applique un traitement sur chacune des primitives reçues. Il est possible d'annuler le traitement de certaines primitives ou d'en créer de nouvelles.

Fragment shader situé après l'étape de rastérisation, il applique un traitement sur chacun des fragments (pixels) générés par la rastérisation des primitives issues du *geometry shader*.

À l'heure actuelle, de nouvelles architectures graphiques[6] sont apparues et ajoutent une étape programmable supplémentaire à la chaîne de traitements graphiques. Cette nouvelle étape, appelée *tesselation shader*, permet d'effectuer plus simplement la subdivision des primitives sur le GPU. Ce nouveau shader est situé juste avant le *geometry shader*.

Note sur les notions de rendu temps réel - interactif La vitesse est un objectif important des méthodes de rendu et de visualisation. Ces méthodes peuvent généralement être classifiées suivant leur fréquence de rafraîchissement (*i.e.* le nombre d'images qu'elles sont capables de produire en une seconde). Nous avons, pour ce manuscrit, choisit la terminologie suivante : on parle de méthodes temps réel lorsque celles-ci sont capables d'effectuer le rendu à plus de 25 images par secondes. Les méthodes qualifiées d'interactives sont les

6. Processeurs de type NVIDIA Fermi.

méthodes permettant encore à l'utilisateur de naviguer interactivement dans la scène. On considère qu'elles doivent atteindre un taux de rafraîchissement compris entre quelques images par seconde et 25 images par secondes. En dessous de cela, on considère ces méthodes comme étant des méthodes de rendu pré-calculés.

3 Objectifs des travaux

Les données à deux dimensions et demie, et leur représentation via des champs de hauteurs, introduisent un certain nombre de difficultés quant à leur visualisation. Le problème principal réside dans la génération d'une surface continue à partir de l'échantillonnage des élévations qui compose la carte de hauteurs. À l'heure actuelle, et comme nous le discutons dans l'état de l'art, aux chapitres 3 et 6, un maillage est généralement reconstruit à partir de la carte de hauteurs. Les méthodes ainsi basées sur des maillages produisent des résultats assez satisfaisants, en termes de qualité et de performances, mais se montrent globalement complexes et très lourdes à mettre en place en ayant, par exemple, recours à de longues phases de pré-calculs. De plus ces différentes méthodes manquent de flexibilité et se retrouvent très souvent cantonnées à des usages bien spécifiques. Une autre approche, permettant de visualiser ces données, consiste à utiliser des algorithmes de lancer de rayons. Historiquement, ces techniques font également partie des premières employées pour rendre des cartes d'élévations mais les calculs qu'elles requièrent, ne les destinaient pas à des applications de rendu temps réel. Cet état a changé avec l'apparition des matériels graphiques actuels, ce qui permet désormais à ces méthodes un début d'utilisation en temps réel. Cependant, cette catégorie de méthodes reste moins performante que celles faisant l'usage de maillages, bien qu'elles présentent certains avantages, comme un bon degré de flexibilité et la possibilité d'obtenir des rendus de très haute qualité, pour des vitesses d'affichage qui demeurent tout à fait convenables.

Quelque soit la méthode de rendu employée, les problèmes restent souvent identiques et sont en partie liés à l'échantillonnage régulier des données, leur taille et le choix d'un bon ratio entre qualité et vitesse de rendu. L'échantillonnage régulier des données, bien que simplifiant grandement leur représentation en mémoire, peut se révéler problématique sur de grandes masses de données, en introduisant des problèmes d'aliasing et de moirés. De plus, la visualisation de ces jeux de données peut présenter de fortes occlusions, mais qui ne peuvent être que difficilement exploitées pour limiter le rendu des données non visibles. Cette masse de données est également une contrainte très forte, qui limite généralement la vitesse de rendu et impose, pour y remédier, d'utiliser

des techniques d'accélération qui reposent, soit sur des algorithmes de simplifications, soit sur des techniques de compressions des données. Ces grandes quantités de données imposent, en plus, des mécanismes pour gérer le plus finement possible la mémoire, afin de ne charger que les données utiles à la vue courante.

Un autre point critique de ces méthodes de visualisation concerne le ratio entre qualité et vitesse du rendu. En effet, tout dépend de l'application visée pour trouver le bon équilibre des performances de rendu. En effet, certaines applications (notamment pour la visualisation scientifique) nécessitent une très grande qualité de rendu mais peuvent admettre une certaine dégradation des performances, tandis que pour d'autres applications (telles que les jeux vidéos), la vitesse d'affichage est primordiale, mais on peut se limiter à une précision moindre du rendu. Le choix de cet équilibre est donc extrêmement subjectif, en dépendant des applications visées, mais doit pouvoir être ajusté facilement, d'où la nécessité d'un certain degré de flexibilité pour ces méthodes de visualisation.

D'autres points critiques sont également à prendre en compte. Parmi ceux là, on compte l'adaptation aux données ciblées qui doit être la meilleure possible pour envisager une méthode de visualisation efficace. Par exemple dans le cas où les jeux de données peuvent être animés et varier au cours du temps. Cela pose des contraintes supplémentaires à la méthode de visualisation qui doit les prendre en charge. Dans d'autres cas, les données ne représentent que des reliefs de faibles amplitudes, par rapport à la taille de l'objet. Il est possible de s'appuyer sur cette propriété pour en accélérer le rendu. C'est la raison pour laquelle, dans cette thèse sur la visualisation de données à deux dimensions et demie, nous avons opté pour une séparation entre la visualisation de reliefs de faibles amplitudes (relief à une échelle millimétrique, équivalent, au plus, à quelques millimètres) et celle de reliefs de fortes amplitudes (reliefs à une échelle macroscopique, de quelques centimètres jusqu'à plusieurs kilomètres). Cette séparation peut également se justifier par les occlusions générées par les données. Dans le premier cas, celles-ci sont généralement faibles, en ayant un impact limité au niveau géométrique, mais toutefois marqué au niveau de l'illumination. Dans le second cas, elles sont très marquées et influent grandement sur le résultat géométrique final de la visualisation. On doit donc considérer séparément les différentes échelles de reliefs représentés par des cartes d'élévations.

L'objectif est de fournir des méthodes de visualisation simples mais offrant une grande qualité de rendu pour des performances temps réel avec un grand degré de flexibilité. L'adaptation ciblées des méthodes aux données implique l'utilisation d'approches différentes. C'est pourquoi la première partie

de ces travaux portent sur la visualisation de données, issues de la numérisation d'objets réels, dont les reliefs sont de faibles amplitudes. Pour illustrer ce problème sur un cas concret, cette méthode est appliquée à la visualisation de tableaux d'art issus d'une étape de numérisation. La seconde partie s'attelle à la visualisation de données où les reliefs sont d'amplitudes plus importantes, mais qui peuvent varier au cours du temps.

4 Contributions

Ces travaux ont débouché sur un certain nombre de contributions qui sont résumés dans cette section. D'une façon globale, les méthodes de visualisation développées apportent une certaine facilité de mise en place et surtout une grande flexibilité dans leur emploi.

En ce qui concerne la visualisation de reliefs de faibles amplitudes, nous avons développé une méthode hybride simple qui combine une simulation des reliefs et le rendu de ceux-ci. Elle offre un haut niveau de performances avec une qualité élevée du rendu. De plus, afin d'obtenir les données, nous avons développé tout un protocole et tous les outils associés, permettant la numérisation et le traitement des données afin de les visualiser. Ainsi, nous avons construit une chaîne de traitements complète, mais qui reste simple à mettre en œuvre, couvrant toutes les étapes depuis la numérisation de tableaux d'art jusqu'à leur visualisation. La numérisation inclut à la fois l'acquisition de la géométrie des toiles (et plus spécialement la couche picturale) et l'information de réflectance. Une contribution majeure de ces travaux réside dans le fait que la géométrie est prise en compte aussi bien durant la numérisation que durant le rendu des toiles, pour en accroître le réalisme. Concernant la méthode de rendu, ces travaux apportent une combinaison efficace entre le relief et le champ de réflectance durant la phase de rendu, permettant d'améliorer le rendu de la réflectance.

Dans le cas de données présentant des reliefs dont l'échelle est plus importante, nous avons également proposé une méthode hybride, simple et flexible, pour les visualiser. Cette méthode a pour caractéristique principale de combiner un algorithme de lancer de rayons avec une représentation du terrain par un maillage. Son originalité réside dans la combinaison de ces deux méthodes qui permet de fournir une adaptation du rendu aux données visualisées, mais sans introduire de mécanismes compliqués à mettre en place et ne nécessitant aucun pré-calcul complexe. Ainsi, cette méthode de visualisation s'applique à des données pouvant évoluer dynamiquement au cours du rendu. Elle obtient une haute qualité de rendu en limitant les traitements qui leurs sont géné-

ralement appliqués et qui altèrent les données, en introduisant par exemple un lissage de leur surface. Ces résultats sont obtenus tout en maintenant des performances de rendu temps réel.

5 Organisation du mémoire

Ce mémoire s'organise en suivant la trame chronologique des travaux menés lors de cette thèse : visualisation de reliefs de faibles amplitudes, puis visualisation de reliefs à fortes amplitudes.

5.1 Déroulement des travaux

Les travaux présentés dans ce manuscrit se sont déroulés en deux étapes distinctes. Dans un premier temps, nous avons souhaité nous concentrer uniquement sur des données formant des reliefs où les amplitudes sont faibles. Cela permet de simplifier le problème en réduisant notamment les auto-occlusions générées par les données. Grâce à cette simplification, nous avons pu tester l'approche hybride qui consiste à combiner les deux techniques de rendu différentes, retenues pour la visualisation de données à deux dimensions et demie.

Une fois l'approche hybride validée sur ce cas concret, nous avons généralisé cette approche à des reliefs plus importants, en prenant comme exemple d'application principal la visualisation de terrains. Ces travaux ont été menés en s'appuyant sur les tests effectués pour développer la méthode hybride de rendu des reliefs de faibles amplitudes, mais en gardant pour objectif la visualisation de données dynamiques. Étant donné l'importance des reliefs, les méthodes de rendu retenues ont été adaptées par rapport au premier cas, mais le principe général de combinaison demeure le même.

5.2 Plan du mémoire

Après ce premier chapitre introductif, cette thèse s'articule autour de deux parties distinctes. La première partie concerne la numérisation et la visualisation de tableaux. Au cours du chapitre 3, après avoir introduit plus en détails les objectifs spécifiques de ces travaux, nous présentons un état de l'art de différentes techniques développées pour ajouter, visualiser et rendre les détails, modélisés à partir de données à deux dimensions et demie, de la surface d'un objet. La méthode, développée pour visualiser la surface des peintures numérisées, est présentée durant le chapitre 4. La chaîne de traitements

permettant la numérisation de ces tableaux est ensuite détaillée au cours du chapitre 5.

La deuxième partie de ce manuscrit s'intéresse à la description de notre méthode hybride de visualisation de données à deux dimensions et demie, dans le cas de reliefs présentant des amplitudes plus importantes que sur la peinture des tableaux. Le chapitre 6 dresse un état des lieux détaillé des méthodes de visualisation de données à deux dimensions et demie, en se focalisant plus particulièrement sur celles dédiées aux données topographiques. La démarche ayant conduit le développement de cette méthode ainsi que son fonctionnement sont décrits en détails au chapitre 7. Les résultats produits par cette méthode sont analysés précisément lors du chapitre 8.

Ce mémoire se termine par une conclusion générale, au chapitre 9, qui dresse un bilan des différentes contributions de cette thèse avant de proposer quelques pistes pouvant donner lieu à des travaux futurs.

Première partie

Visualisation de reliefs de faibles amplitudes

—

Application à la numérisation et à la visualisation temps réel de peintures d'art

Chapitre 2

Présentation de la numérisation et visualisation de peintures d'art

Au cours de cette première partie, nous décrivons la méthode développée pour visualiser des données à deux dimensions et demie, dont les reliefs présentent une échelle faible. Comme cadre applicatif, nous avons choisi d'appliquer cette méthode à la numérisation et à la visualisation, de façon réaliste et en temps réel des peintures d'art. Dans ce chapitre, nous présentons plus spécialement les objectifs de cette méthode ainsi que les méthodes existantes liées à la numérisation de telles pièces.

1 Visualisation de reliefs de faibles amplitudes : contexte et objectifs

Le but des travaux effectués dans le cadre de cette thèse est de visualiser des données à deux dimensions et demie. Cependant, comme nous l'avons mentionné dans le chapitre introductif, la méthode de visualisation doit être développée pour s'adapter au mieux aux données ciblées par l'application. C'est la raison pour laquelle nous nous intéressons, dans cette partie, à visualiser des données qui présentent de faibles amplitudes dans leurs reliefs. Comme nous l'avons déjà précisé dans l'introduction de cette thèse, nous considérons que des reliefs d'un champ de hauteurs sont de faible amplitude lorsqu'ils ne représentent qu'une faible hauteur en comparaison à la taille de objet dont il définissent la surface (jusqu'à quelques millimètres pour un tableau représentant une surface de 50 à 100 cm^2). Le but est donc, dans cette première

partie, de proposer une méthode efficace pour la visualisation de telles données, qui présentent certaines caractéristiques particulières, comme de faibles auto-occlusions et un support de base planaire. Il existe un grand nombre d'exemples, d'objets pouvant être représentés par un champ de hauteurs qui forment de faibles reliefs, comme la surface crépie, régulière ou non, d'un mur, la surface d'un terrain (à une échelle très proche de sa surface) parsemée de cailloux ou présentant des creux et de bosses, etc. Toutefois, pour développer cette méthode et afin de limiter les différents cas particuliers d'utilisations, nous avons choisi un cadre applicatif très précis, lié à l'héritage culturel.

L'héritage culturel est un domaine nécessitant de plus en plus fréquemment des outils permettant l'acquisition d'objets réels à des fins de sauvegarde, d'étude ou encore de diffusion. En effet, l'utilisation de copies numériques en lieu et place des objets réels permet de faciliter la diffusion de ces objets qu'il est parfois difficile, voire impossible de déplacer, car trop fragiles ou trop volumineux. Cela permet également une diffusion au plus grand nombre de ces œuvres d'art ou pièces historiques, au travers de galeries virtuelles accessibles, par exemple, sur Internet. C'est la raison pour laquelle, un certain nombre de dispositifs matériels et de systèmes logiciels ont été proposés en vue de numériser ces différents objets d'art ou reliques historiques. Parallèlement à ces développements et avec pour objectif de les tester, des campagnes de numérisations ont vu le jour pour acquérir numériquement certains objets historiques marquants. Par exemple, depuis quelques années le projet Michelangelo, initié par Levoy *et al.* [LPC+00], s'est fixé pour objectif de construire une base de données de maillages issus de la numérisation des principales sculptures de Michel-Ange.

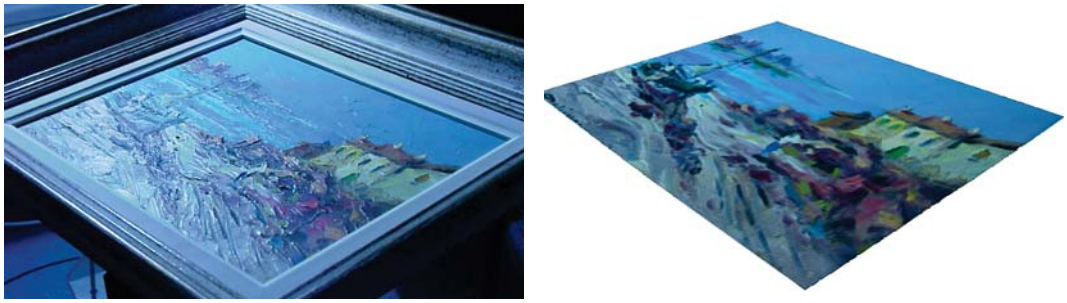
Les dispositifs de numérisation en trois dimensions, mis au point jusqu'à présent, fournissent des modèles géométriques de plus en plus complexes et détaillés, qui nécessitent très souvent des techniques dédiées pour leur manipulation et leur visualisation. Étant donné la grande diversité des objets susceptibles d'être numérisés, il est pertinent de se demander s'il ne serait pas utile, dans un soucis d'efficacité, aussi bien pour le processus de numérisation proprement dit que pour l'exploitation de ces données, de développer des techniques propres à chaque type d'objets. Le cas des peintures d'art nous est apparu intéressant à étudier. En effet, si on observe la surface d'un tableau, on constate que, suivant la technique employée par l'artiste, la peinture forme une surface particulière sur la toile, comme illustré par les exemples réels par la figure 2.1. Cette surface est composée d'un certain nombre de reliefs, à l'échelle macroscopique, qui jouent un rôle important dans les interactions entre la lumière et le tableau. Dans certains cas, pour les peintures au couteau ou dans certaines œuvres plus contemporaines par exemple, il se peut que ces reliefs fassent partie de la démarche artistique ayant menée à la création du tableau.

Figure 2.1 – Exemples de reliefs formés par la peinture de différents tableaux.

Il nous est donc apparu important de prendre en compte cette spécificité dans le cadre de la numérisation et le rendu de tels objets.

Cependant, il faut souligner que les reliefs qui forment la couche picturale, ont généralement une amplitude assez faible (de l'ordre de quelques millimètres), ce qui permet de les représenter de façon assez fidèle en utilisant un simple champ de hauteurs. Ce type de représentation offre également un avantage non négligeable durant l'étape d'acquisition des données. En effet, les matériels employés pour effectuer une numérisation à trois dimensions utilisent généralement des représentations assez proches des champs de hauteurs pour fournir les résultats d'une acquisition. La représentation au moyen d'une carte d'élévations peut ensuite être directement employée pour ajouter des détails géométriques à la surface d'un objet, comme le montrent les méthodes décrites dans le chapitre 3.

L'autre élément important dans la numérisation d'une peinture est l'acquisition de la couleur. En effet, celle-ci représente la caractéristique principale d'une toile et doit être fidèlement numérisée pour être restituée. Cependant, comme l'illustre la figure 2.2, cette information ne doit pas se limiter à une simple couleur mais doit prendre en compte la réflectance de la surface qui, pour un même point de la peinture, engendre des changements de couleurs



(a) Photographie réelle d'un tableau

(b) Rendu du tableau numérisé avec une texture de couleurs diffuses

Figure 2.2 – Comparaison entre une photographie d'un tableau et le rendu correspondant à sa numérisation, pour une direction d'éclairage proche, illustrant l'importance de la réflectance pour le rendu de la peinture.

en fonction des directions d'observation et d'éclairage. Effectivement, la réflectance permet de caractériser les réflexions lumineuses engendrées par le matériau qui compose la surface d'un objet (en l'occurrence dans ce cas la peinture de la toile). Par conséquent, elle modifie notre perception de la couleur de la toile en fonction, en particulier, de son éclairage. Pour numériser cette information et la représenter, un certain nombre de techniques existent et sont présentées à la section 3.

Ces travaux ont donné lieu à quelques contributions. Nous avons, en effet, développé une méthode hybride de visualisation temps réel et réaliste de données à deux dimensions et demie présentant des reliefs de faibles amplitudes. En parallèle de cette méthode de visualisation, nous avons proposé une chaîne de traitements originale qui fournit un protocole complet de numérisation géométrique et colorimétrique de tableaux, ainsi qu'un certain nombre d'outils simples à mettre en œuvre et à utiliser permettant de traiter les données en vue de leur visualisation. Ces deux contributions ont été validées par l'étude d'un cas pratique, à savoir la numérisation d'une toile.

2 Numérisation de tableaux : les antécédents

L'état de l'art sur la numérisation de tableaux à des fins de visualisation reste assez pauvre : les travaux étant plus orientés dans une optique d'étude des peintures et se concentre, par conséquent, plus sur une acquisition à haute définition des tableaux. De manière générale, la géométrie n'est approximée que par un simple plan et l'information chromatique est capturée par un ensemble de photographies multi-spectrales. En effet, comme le décrivent Martinez *et al.*

[MCSP02], un certain nombre de travaux ont été menés pour développer de tels dispositifs de numérisation. Ces dispositifs se focalisent principalement sur une acquisition multi-spectrales à très haute résolution des peintures [PRPC09].

En ce qui concerne la géométrie de la peinture quelques travaux en tiennent compte, mais uniquement pendant la phase de numérisation. Parmi ces différents travaux, nous pouvons citer ceux de Tominaga *et al.* [TTK04], qui ont développé une technique permettant l'acquisition de la couleur d'une peinture à l'aide d'une caméra multi-bandes. Dans ces travaux, le relief de la surface n'est considéré qu'au moment de l'acquisition : la rugosité n'est utilisée que pour déterminer la réflectance spectrale de la peinture et n'est pas prise en compte durant la phase de rendu.

Les travaux de Grattoni et Spertino [GS03] présentent une méthode pour numériser des surfaces peintes, en termes de géométrie et de couleurs, à l'aide d'un outil optique. Cependant, la géométrie considérée est celle du support de la peinture et non celle de la peinture elle même. En effet, l'objectif de ces travaux est de numériser des surfaces, telles que des fresques, dans un soucis de conservation et de restauration.

Toujours avec pour objectif premier la restauration des œuvres d'art, Fontana *et al.* [FGG+05] présentent un dispositif complet de numérisation de peintures qui permet non seulement une acquisition multi-spectrale de la peinture mais intègre également une numérisation en trois dimensions de la surface de la toile. Cependant, ces données ne sont destinées qu'à être exploitées en vue de l'étude de la peinture. Dans la même optique, Guidi *et al.* [GAL03] expliquent comment ils numérisent, à l'aide d'une caméra permettant une acquisition en trois dimensions, une peinture sur bois de Léonard De Vinci. La géométrie est utilisée, dans ce cas précis, afin de fournir un diagnostique sur l'état de conservation du support en bois de la peinture, mais ne considère pas le relief à des fins de visualisation. De façon similaire, on peut encore citer Akça *et al.* [AGBL07] qui décrivent l'utilisation de techniques de numérisation en trois dimensions appliquées pour la numérisation d'un tableau.

Toutes ces méthodes ont principalement été développées pour des besoins liées à l'étude des œuvres d'art. Il en résulte des techniques s'appuyant sur des dispositifs matériels d'acquisition des données complexes qui permettent cependant de générer des données à très haute définition. Néanmoins, les contraintes liées à la visualisation temps réel de ces données ne sont pas prises en compte par ces différentes techniques de numérisation. De plus, l'information géométrique représentant le relief de la peinture, qui est de plus en plus fréquemment capturée, n'est pas exploitée, dans le cadre de la visualisation, et n'est, dès lors, pas intégrée au rendu de l'image.

3 Acquisition et rendu de la réflectance d'une surface

Les travaux ayant trait à la numérisation de peintures d'art ne prennent également que peu en compte la réflectance de la peinture. En effet, les changements de couleurs sur la surface sont liés à la variation spatiale de la réflectance induite par les différentes peintures utilisées par l'artiste. De nombreux travaux se sont attachés à la mesure et au rendu de la réflectance de surfaces.

Afin de modéliser ce comportement bi-directionnel de la réflectance, on utilise généralement des modèles statistiques. Ces modèles sont définis par des fonctions, comme par exemple la fonction de réflectance bi-directionnelle (*bidirectionnal réflectance distribution fonction, BRDF*) introduite, dans le domaine de l'informatique graphique, par Cook et Torrance [CT81]. Cette fonction permet d'encoder la couleur d'un matériaux en fonction du point de vue et de la position des sources lumineuses. Les données utilisées pour générer ces différentes fonctions peuvent être issues d'un processus de numérisation d'objets réels [WLL+09]. Une *BRDF* ne peut modéliser que le comportement lumineux d'un matériaux en un seul point précis de la surface d'un objet. Toutefois, sur n'importe quel objet, et en particulier dans le cas des peintures, les caractéristiques du matériaux qui le composent, varient spatialement. Pour modéliser ces variations de *BRDF* sur une surface, différent travaux ont été menés, notamment en développant des fonctions de réflectance bi-directionnelle variant spatialement [LKG+03].

Ces différentes variations incluent les changements des propriétés lumineuses du matériau composant l'objet. Elles sont généralement dues aux micro-reliefs qui composent la surface d'un objet et qui contribuent donc aux interactions lumineuses en générant, en particulier, des auto-réflections et des auto-ombrages. Ces différents phénomènes sont importants et doivent être pris en compte finement pour un rendu réaliste de la réflectance. Toutefois, il faut distinguer plusieurs cas. En effet, les *BRDF* ne prennent en compte que le reliefs d'échelle très fine et non visible à l'œil nu. Cependant, dans de nombreux cas (écorces d'arbres, crépis rugueux, mais également peintures) ces reliefs sont de taille plus importante et deviennent visibles. Il faut donc les intégrer aux modèles de représentation de la réflectance. C'est pourquoi ont été introduites les fonctions de texture bi-directionnelle (*bidirectionnal texture function*, *BTF*) par Dana *et al.* [DNGK97, DGNK99]. Ces fonctions encodent donc les variations spatiales de réflectance en tenant compte, contrairement aux *BRDF* variant spatialement, des reliefs visibles à l'échelle humaine de la surface du matériau. Une contrainte majeure de cette représentation réside dans la phase d'acquisition de ces données qui nécessitent un matériel très complexe pour obtenir des données suffisamment précises pour être exploitées. Une seconde

contrainte forte de ces représentations se situe au niveau de leur stockage. En effet, comme les fonctions de texture bi-directionnelle représentent des fonctions à six dimensions, elles nécessitent de grandes quantités de mémoire pour être stockées. Cette contrainte implique que ces fonctions sont assez peu adaptées, en l'état, au rendu temps réel et leur utilisation est, pour ces raisons, très souvent limitée à des échantillons de taille très réduite.

La taille de ces représentations est donc problématique. Par conséquent, des travaux ont été menés afin de proposer des méthodes destinées à compresser les données représentant, en particulier, les *BRDF* ainsi acquises [LFTG97, MWL+99, NDM05]. Un modèle de représentation assez commun est, par exemple, celui introduit par Lafortune *et al.* [LFTG97] qui proposent de représenter la réflectance par un ensemble de lobes spéculaires. Afin de les encoder, il utilise une somme de termes qui définissent la forme des différents lobes spéculaires, comme le montre l'équation 2.1.

$$f_r(\vec{v},\vec{l}) = \rho_d + \sum_i \rho_{s,i} \cdot s_i(\vec{v},\vec{l}) \tag{2.1}$$

Sur cette équation, $\vec{v}$ représente le vecteur de vue et $\vec{l}$ la direction d'éclairage. Les coefficients ρ_d et ρ_s représentent respectivement la refléctance diffuse et l'albédo correspondant aux reflets spéculaires. Ce dernier est pondéré suivant le terme s définissant la forme d'un lobe qui peut être exprimé suivant cette équation :

$$s(\vec{v},\vec{l}) = \left(\begin{bmatrix} \vec{l}_x \\ \vec{l}_y \\ \vec{l}_z \end{bmatrix}^T \cdot \begin{bmatrix} C_x & & \\ & C_y & \\ & & C_z \end{bmatrix} \cdot \begin{bmatrix} \vec{v}_x \\ \vec{v}_y \\ \vec{v}_z \end{bmatrix} \right)^k$$

Les coefficients C_x, C_y et C_z permettent de paramétrer la forme des lobes spéculaires. Pour de plus amples détails sur ces différentes techniques, un état de l'art des fonctions de texture bi-directionnelle a été publié par Müller *et al.* [MMS+05].

Comme cette représentation de la réflectance est bien adaptée aux processeurs graphiques et permet de résultats visuellement convaincant sans surcoûts trop importants de calculs, à l'instar de McAllister *et al.* [MLH02], notre méthode utilise un modèle de Lafortune pour représenter la réflectance en chaque point de la surface du tableau. Comme l'a souligné Meseth *et al.* [MMK03], la mesure de *BTF* pour des surfaces dont les reliefs sont importants, introduit le plus souvent de grandes imprécisions. En effet, celles-ci sont introduites par les auto-occlusions qui deviennent trop fortes et trop présentes dans ces cas. Elles sont ainsi difficiles à prendre en compte par les modèles mathématiques employés pour construire les *BTF* sans multiplier fortement les échantillons lumineux effectués lors de la capture de réflectance. C'est pourquoi notre méthode se concentre à la fois sur l'acquisition et le rendu du relief de la peinture,

au lieu de n'utiliser qu'une *BTF*. On peut ainsi découpler les données représentant le relief de celles encodant la réflectance, pour supprimer ces artéfacts.

La plupart des techniques précédemment citées sont, sous certaines conditions, peu adaptées à un rendu temps réel et réaliste, du fait de leur coût en temps de calculs (dûs à la masse de données, aux algorithmes de compression et de codage nécessaire, ...) ou de leur qualité de rendu inadéquate (due à la compression des données ou aux approximations faites). Les méthodes hybrides peuvent alors fournir une alternative intéressante. De tels mécanismes consistent à utiliser des algorithmes de rendu différents en fonction de certains paramètres, comme la position relative du point de vue ou la taille de l'objet projeté à l'écran. Becker *et al.* [BM93], par exemple, utilisent alternativement trois algorithmes différents, selon le point de vue et sa distance à la scène. Parmi ces trois algorithmes, outre une *BRDF*, Becker *et al.* utilisent deux techniques permettant d'ajouter des détails à la surface d'un objet. Ces méthodes utilisent comme information géométrique pour modéliser les détails, des données qui sont à la base des données à deux dimensions et demie.

Étant donnée la nature des reliefs qu'il faut visualiser, et leur représentation de base, nous avons également opté, pour visualiser une peinture numérisée, d'utiliser de telles méthodes d'ajout de détails. C'est pourquoi, un certain nombre de ces méthodes d'ajout de détails, utilisant des champs de hauteurs, font l'objet d'une étude détaillée au chapitre suivant.

Chapitre 3

Ajout de détails à partir de champs de hauteurs

Dans ce chapitre, nous décrivons un état de l'art des méthodes permettant d'ajouter des détails géométriques supplémentaires à des objets afin de les enrichir visuellement. Nous nous concentrons exclusivement sur les méthodes où les détails ajoutés peuvent être représentés par des données à deux dimensions et demie.

1 Généralités sur les techniques d'ajout de détails

Comme mentionné dans le chapitre introductif de cette thèse ainsi que dans le chapitre précédent, les données à deux dimensions et demie ont un vaste champ d'applications allant de la représentation de données topographiques à la représentation de détails fins d'une surface d'objets complexes. En effet, ce type de données peut également servir à la modélisation de détails géométriques : un objet modélisé avec un faible niveau de précision est alors enrichi visuellement à l'aide de détails plaqués sur sa surface au moment du rendu. L'ajout de détails se justifie par la complexité grandissante des objets virtuels manipulés dans les applications graphiques. Ces objets sont principalement représentés par des maillages et la masse de données qu'ils génèrent est bien souvent difficile à gérer pour les architectures graphiques [1]. Par conséquent, il devient de plus en plus difficile de visualiser une multitude d'objets de ce type tout en maintenant un rendu temps-réel.

1. La difficulté à afficher de tels objets est, bien évidemment, dépendante du matériel graphique employé, mais, généralement, la complexité des objets augmentent parallèlement à la puissance du matériel graphique.

La contrainte se situe généralement au niveau matériel. Les maillages qui forment ces objets, sont composés d'un très grand nombre de primitives géométriques (jusqu'à plusieurs dizaines de millions) qu'il faut stocker et rastériser. Malgré le support matériel offert par les processeurs graphiques, leur visualisation reste coûteuse, en termes d'occupation mémoire et de temps de calculs : transfert des données vers le processeur graphique, projection des sommets en espace image, calculs d'illumination pour l'ensemble des fragments générés par la rastérisation, etc. Il est donc nécessaire de réduire le nombre de primitives géométriques traitées afin d'obtenir des performances de rendu plus élevées. Par conséquent, l'objectif principal des méthodes d'ajout de détails est d'accroître la vitesse de rendu d'objets complexes en réduisant fortement le nombre primitives envoyées à la carte graphique pour rastérisation.

Afin d'y parvenir, ces méthodes s'appuient sur une décomposition de la surface des objets. En effet, la surface d'un objet peut être vue suivant plusieurs niveaux d'échelle, en fonction de l'information géométrique que l'on considère. On dégage généralement trois niveaux différents : la *macrostructure*, la *mésostructure* et la *microstructure*. La macrostructure représente l'apparence de l'objet dans sa globalité en définissant sa forme générale. La mésostructure s'apparente à une structure plus fine de l'objet mais encore perceptible à l'œil nu : elle forme les détails géométriques de la macrostructure. C'est ce niveau de détails que s'attachent à modéliser et à rendre les méthodes décrites dans cette section. La microstructure représente le niveau de modélisation le plus fin. Elle s'apparente davantage aux propriétés du matériau qui compose l'objet, comme les micro-facettes, invisibles à l'œil nu, qui conditionnent une grande partie du comportement lumineux de l'objet.

Pour modéliser ces niveaux, différentes représentations sont employées. La macrostructure est généralement représentée par un maillage. Étant donné que la macrostructure ne représente que la forme générale de l'objet, son maillage n'est pas nécessairement très fin. À l'autre extrémité du spectre des échelles et de part les propriétés qu'elle modélise, la microstructure fait intervenir des structures de données et des techniques particulières pour sa représentation. On emploie fréquemment des modèles statistiques, basées sur des distributions de micro-facettes, qui permettent de modéliser les interactions qui sont générés entre la lumière et la matière. Cette modélisation est formalisée via différentes fonctions, comme la fonction de réflectance bi-directionnelle (*bidirectionnal reflectance distribution fonction, BRDF* [CT81, HTSG91, KE09]) ou la fonction de distribution de reflection et de diffusion bi-directionnelle (*bidirectional surface scattering reflection distribution function, BSSRDF*) [JMLH01]. En ce qui concerne la mésostructure, la représenter par un maillage implique que celui-ci soit composé de nombreux polygones, ce qui serait pénalisant en terme de performances et d'occupation mémoire, au regard de ce que nous

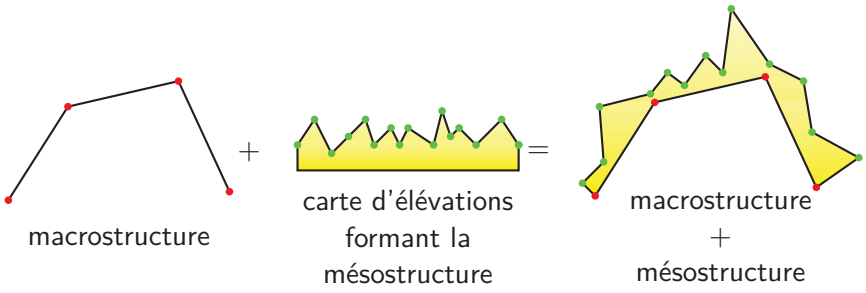


Figure 3.1 – Principe de fonctionnement des méthodes d'ajout de détails : combinaison d'une macrostructure et d'une carte d'élévations permettant d'ajouter une mésostructure à l'objet.

avons dit précédemment. Par conséquent, les méthodes d'ajout de détails décrites dans cette section, s'attachent à exploiter une autre forme de représentation : des données à deux dimensions et demie. Ce type de données est en effet bien adapté pour représenter des reliefs de faibles amplitudes (caractéristiques de ces types de détails) de façon compacte et exploitables sans avoir recours à des mécanismes trop complexes à implanter.

Il faut toutefois noter qu'il existe d'autres mécanismes permettant d'ajouter des détails à la surface d'objets. On peut citer, à titre d'exemple, les textures volumétriques [KK89]. Néanmoins, comme ces différentes techniques s'appuyent sur des représentations des détails beaucoup plus complexes que de simple données à deux dimensions et demie, elle ne seront pas traitées dans cet état de l'art.

Parmi toutes les méthodes d'ajout de détails basées sur des champs de hauteurs, on peut distinguer deux approches différentes pour représenter et visualiser la mésostructure : celles où le relief des détails n'est que simulé et celles où la géométrie des détails est réellement ajoutée. Néanmoins, le fonctionnement de ces méthodes suit le même principe : les détails sont plaqués sur la macrostructure en suivant une carte d'élévations représentant la mésostructure, tel que décrit par la figure 3.1.

La première catégorie de méthodes pose l'hypothèse que les reliefs de la mésostructure sont faibles et qu'ils peuvent être simplement simulés. Ces méthodes reposent essentiellement sur une perturbation du calcul d'illumination, via des informations (vecteurs normaux notamment) extraites de la cartes d'élévations. Dans le cas des méthodes de la seconde catégorie, la géométrie de la mésostructure est ajoutée, à la volée, durant le rendu. Pour cela, ces méthodes appliquent directement des modifications à la géométrie de la macrostructure, en suivant la carte d'élévations. Ces transformations sont appliquées soit en effectuant les modifications de la géométrie directement sur

le maillage, soit en déformant la géométrie via un traitement par pixels. Dans le premier cas, on parle de placage de déplacements par sommets. Ces méthodes nécessitent de disposer d'un maillage suffisamment fin pour obtenir une mésostructure détaillée. Ce maillage peut provenir d'une subdivision de la macrostructure. Dans le second cas, on parle de placage de déplacements par pixels [SKU08]. Ces méthodes travaillent en espace image après la projection et la rastérisation du maillage de la macrostructure. Elles fonctionnent sur le principe suivant : déterminer pour chaque pixel de la macrostructure s'il appartient bien à la mésostructure et, si oui, quelle est sa position réelle sur la mésostructure. Afin de résoudre ce problème, l'approche communément retenue met en œuvre un algorithme de lancer de rayons effectué pour chaque pixel de la macrostructure. Le lancer de rayons peut-être appliqué au moment du rendu, directement sur le processeur graphique, ou être pré-calculé.

Il est à noter, que, pendant très longtemps (*i.e.* jusqu'à l'apparition des cartes graphiques de dernière génération gérant la subdivision directement sur le GPU, *cf.* section 4), les méthodes de placage de déplacements par sommets ont introduit certains problèmes quant à la gestion du maillage utilisé pour appliquer le déplacement. En effet, contrairement aux visées initiales de l'ajout de détails, il fallait ici avoir recours à un maillage très fin qui était, par conséquent, difficile à manipuler et à rendre. Cependant, comme ces méthodes constituent historiquement les précurseurs des méthodes de rendu par pixels, elles sont présentées en détails dans ce chapitre.

Tout au long de ce chapitre, nous présentons donc différentes techniques proposées afin d'ajouter des détails à la macrostructure d'un objet, à partir d'un représentation de ceux-ci par des données à deux dimensions et demie. Dans un premier temps, au cours de la section 2, les méthodes utilisant une simulation visuelle du relief de la mésostructure sont décrites. Dans un second temps, dans les sections 3 et 4, nous décrivons les méthodes modifiant réellement la géométrie des objets et, plus particulièrement, celles basées sur des traitements par pixels aux sections 5, 6 et 7. Pour finir, un bilan de ces différentes méthodes est dressé dans la section 9.

2 Ajout de détails par simulation du relief

Comme nous venons de le décrire, certaines méthodes d'ajout de détails s'appuient sur l'hypothèse que les reliefs de la mésostructure sont faibles par rapport à la macrostructure et que, par conséquent, une simple simulation visuelle de ceux-ci, appliquée par une modification du calcul d'ombrage, est suffisante. La méthode posant les plus fortes hypothèses, et historiquement la

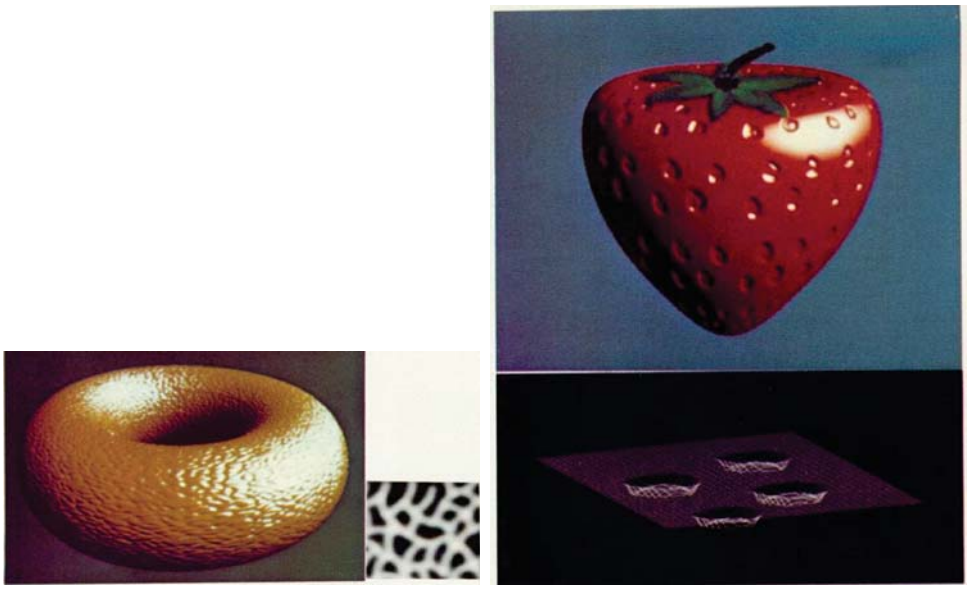


Figure 3.2 – Exemples de *bump mapping* présentés par Blinn, ainsi que les champs de hauteurs dont les vecteurs normaux sont issus. Extrait de [Bli78].

première méthode d'ajout de détails, est le placage de textures [BN76]. Cette technique ne traite pas réellement le problème de la mésostructure. Elle se limite simplement à modifier la couleur locale de la macrostructure. Elle peut s'apparenter à la représentation d'une mésostructure indépendante du point de vue et de l'éclairage.

La première méthode qui s'attache à réellement traiter le problème de la mésostructure et à tenir compte d'une géométrie de petite échelle est le *bump mapping* introduit par Blinn [Bli78]. Le *bump mapping* [Bli78] représente la mésostructure par une variation des vecteurs normaux tout au long des polygones composant la macrostructure. Les vecteurs normaux sont extraits à partir du champ de hauteurs représentant la mésostructure. Ils sont ensuite utilisés lors du calcul d'illumination des différents pixels de la macrostructure. Cette technique produit une simulation des reliefs, comme le montre la figure 3.2.

Cette technique n'ajoute cependant aucune géométrie à la macrostructure, ce qui introduit des erreurs de parallaxe importantes : les auto-occlusions générées par la mésostructure ne sont pas prises en compte. Ces erreurs sont renforcées par l'approximation utilisée pour extraire les normales du champ de hauteurs. En effet, la normale utilisée pour calculer l'illumination d'un pixel donné est celle qui correspond au point de la macrostructure associé à ce pixel courant : en se référant à la figure 3.3, la normale $\overrightarrow{n_1'}$ est utilisée au lieu de la normale $\overrightarrow{n_2'}$. De plus, un autre problème de cette approche est que la silhouette

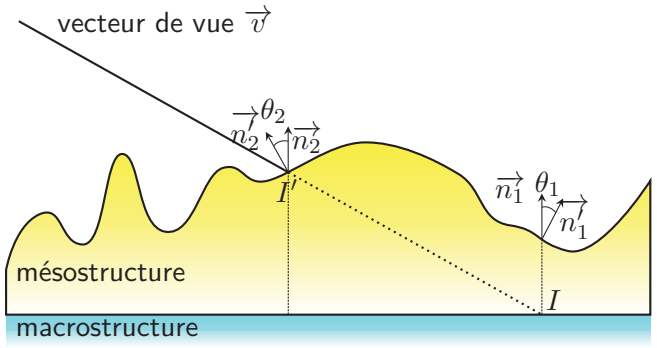


Figure 3.3 – Schéma de fonctionnement du *bump mapping* pour un objet observé suivant la direction $\overrightarrow{v}$. Lors du calcul d'illumination, le vecteur normal employé correspond au vecteur $\overrightarrow{n'_1}$ obtenu en appliquant une rotation d'angle θ_1 au vecteur normal $\overrightarrow{n_1}$.

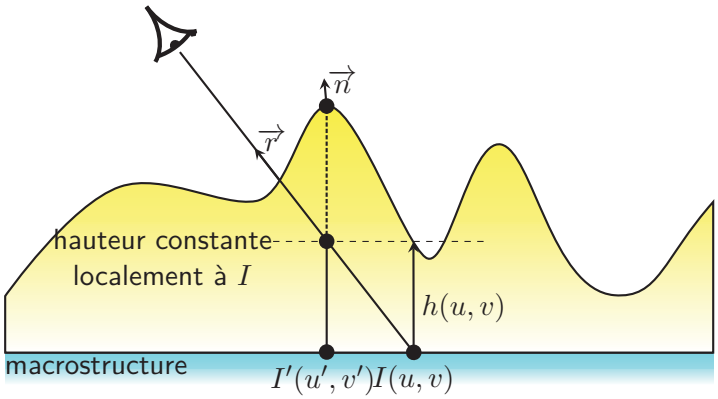


Figure 3.4 – Principe du *parallax mapping* : le calcul de la normale est effectué après l'application d'un décalage sur les coordonnées de textures qui est fonction de la hauteur h du champ hauteurs au pixel courant.

des objets reste celle de la macrostructure, c'est-à-dire une surface modélisée avec un maillage plus grossier.

En dépit des approximations faites, cette méthode est fréquemment employée car elle peut facilement être implantée sur le processeur graphique. Dans ce cas, son coût d'exécution se montre négligeable (un seul accès texture pour calculer la normale). Cependant, pour en faciliter l'implantation GPU, Peercy *et al.* [PAC97] proposent de pré-calculer des cartes de normales à partir des champs de hauteurs et de les stocker dans des textures couleurs RGB. Cette méthode est connue sous le nom de *normal mapping* et reprend exactement le même fonctionnement que le *bump mapping*.

Pour corriger certains problèmes du *bump mapping* et en augmenter le réalisme, le *parallax mapping*, introduit par Kaneko *et al.* [KTI+01], est basé

sur le constat suivant : le *bump mapping* utilise le champ de hauteurs uniquement pour calculer la normale à partir de la macrostructure. Il n'utilise pas l'information géométrique de mésostructure contenue dans le champ de hauteurs. Le *parallax mapping* propose d'améliorer le *bump mapping* en exploitant la géométrie de la mésostructure. Les auteurs proposent de corriger la position utilisée pour interroger le champ de hauteurs du point de la macrostructure associé au pixel courant. Pour cela, ils appliquent un décalage aux coordonnées de textures afin de s'approcher du point de la mésostructure réellement vu pour le pixel courant.

La figure 3.4 présente le principe de fonctionnement du *parallax mapping*. Cette méthode considère que la hauteur h est constante dans le voisinage du point I d'intersection entre le rayon issu du pixel courant et la macrostructure. À partir de h, les coordonnées d'un point I', approximant l'intersection supposée avec la mésostructure, sont déterminées en calculant l'intersection entre le rayon et le plan défini par la hauteur h. Elles sont ensuite utilisées pour calculer le vecteur normal de la mésostructure qui permet, de façon analogue au *bump mapping*, de calculer l'illumination du pixel courant.

Cette méthode donne des résultats assez satisfaisants sur certains types d'objets, tels que les murs de briques. Cependant, son principal défaut apparaît lorsque l'angle de vue devient rasant par rapport à la surface de la macrostructure. Dans ce cas, la valeur du décalage appliqué aux coordonnées de textures tend vers l'infini, ce qui engendre un manque de cohérence au niveau de la couleur des pixels voisins. Cela se traduit visuellement par du bruit.

Afin de minimiser ce problème, Welsh [Wel04] introduit une limite sur la valeur du décalage appliqué aux coordonnées de texture. Pour affiner encore le calcul du décalage, McGuire et McGuire [MM05] proposent d'intégrer la normale de la macrostructure au point d'intersection I au calcul du décalage. Cependant, les vecteurs normaux de la mésostructure ainsi calculés restent toujours approximés plus ou moins fidèlement. Par conséquent, des erreurs de parallaxe restent commises et se traduisent par des artéfacts visuels.

3 Placages de déplacements : principe général

Afin de combler les lacunes des méthodes d'ajouts de détails qui se limitent à la simulation de la mésostructure, Cook [Coo84, CCC87] propose une évolution majeure pour la représentation de la mésostructure, via l'introduction du placage de déplacements (*displacement mapping*). Par cette notion, Cook propose de perturber réellement et directement la géométrie de la ma-

crostructure afin de former la mésostructure : elle est générée, au moment du rendu, à partir de la macrostructure et d'une carte de déplacements. La carte de déplacements encode la distance entre la macrostructure et la mésostructure. Les déplacements sont appliqués suivant la normale des polygones formant la macrostructure et sont exprimés dans l'espace tangent aux polygones. Ce principe est illustré par la figure 3.1.

Plus formellement, si $P(u_P, v_P)$ représente un point de la macrostructure, $\overrightarrow{N}$ le vecteur normal unitaire du polygone courant (*i.e.* vecteur normal de la macrostructure) et $h(u, v)$ la distance du déplacement d'un point, alors les points P' de la mésostructure peuvent-être exprimés par :

$$P' = P + \overrightarrow{N} \cdot h(u_P, v_P)$$

L'utilisation de cartes de déplacements permet de profiter d'un stockage utilisant une texture. On profite ainsi de la cohérence et de la régularité des données qui facilitent leur exploitation. Une paramétrisation du maillage de la macrostructure permet d'associer la carte de déplacements aux polygones de la macrostructure, au même titre que les paramétrisations de surface utilisées pour le placage de textures couleurs.

Dès les premières implantations du placage de déplacements, deux approches distinctes se sont dégagées. La première utilise un déplacement des sommets formant le maillage de la macrostructure. À titre d'exemple, on peut citer les travaux de Miyata [Miy90] permettant de générer et de rendre des murs de pierres. La seconde approche a recours à des algorithmes de lancer de rayons. Patterson *et al.* [PHL91] est le premier à proposer ce type d'algorithmes pour appliquer les déplacements. Ces deux approches sont développées dans les deux sections suivantes.

4 Placages de déplacements par sommets

L'approche la plus simple pour effectuer un placage de déplacements par sommets consiste à déplacer chaque sommet de la macrostructure. Le déplacement est effectué suivant la distance stockée dans la carte de déplacements et sa direction est définie par la normale à la surface. Pour obtenir une mésostructure détaillée suivant ce principe, il est nécessaire de disposer d'un maillage fin pour représenter la macrostructure. Avant l'apparition des architectures graphiques programmables, cette contrainte ne permettait pas à cette approche de fournir des performances de rendu interactif. Il a donc été nécessaire de développer différentes stratégies pour permettre des rendus interactifs. Ces

Entrées : $\overrightarrow{p} \leftarrow$ sommet de la subdivision de la macrostructure ;
$\overrightarrow{n} \leftarrow$ normale de la surface de la macrostructure ;
$\overrightarrow{uv} \leftarrow$ coordonnées de texture associées au sommet;
`h` $(\overrightarrow{x})$: fonction qui, aux coordonnées $\overrightarrow{x}$, associe l'élévation de la carte de hauteurs ;
Sorties : $\overrightarrow{p'}$: nouvelle position du sommet $\overrightarrow{p}$;

$d \leftarrow$`h` $(\overrightarrow{uv})$;
$\overrightarrow{p'} \leftarrow \overrightarrow{p} + \overrightarrow{n} * d$;

Algorithme 3.1 – Algorithme de *vertex displacement mapping* pour un vertex shader d'un GPU actuel : les sommets du maillage sont simplement déplacés suivant la normale de la macrostructure et l'élévation correspondante dans le champ de hauteurs.

méthodes utilisent des techniques de maillages adaptatifs pour simplifier la géométrie rendue tout en maintenant un niveau de détails satisfaisant. Leur objectif est de réduire au maximum le nombre de triangles envoyées à la carte graphique en s'appuyant sur ses capacités d'accélération matérielle [2]. Ces techniques ont été développées dans le même esprit que les techniques de remaillage utilisées pour visualiser des terrains, comme celles décrites au chapitre 6.

L'une des premières implantations dédiée à une accélération par le matériel graphique a été proposée par Gumhold et Hüttner [GH99]. Elle utilise un algorithme de subdivision de maillage, dépendant du point de vue. Cet algorithme permet d'affiner la surface de la macrostructure pour y effectuer le déplacement des sommets, mais uniquement quand cela est nécessaire. Pour réduire encore le nombre de triangles générés lors de la subdivision, d'autres algorithmes de subdivision adaptatifs ont été proposés [DH00, BAD+01, MM02]. Une autre approche, consistant à déplacer les sommets après leur projection en espace image, a également été proposée [DKS01].

Ces méthodes ont permis de disposer d'algorithmes de placage de déplacements s'exécutant suffisamment rapidement pour un rendu interactif mais se sont montrées inadaptées aux générations de GPU programmables. En effet, ces méthodes effectuent l'étape de subdivision sur le CPU et reconstruisent un maillage à chaque image, ce qui s'avère inefficace sur les architectures graphiques actuelles, en partie à cause des nombreux transferts que cela engendre entre la mémoire centrale et la mémoire graphique. Cependant, il est désormais possible d'effectuer l'étape de subdivision directement au niveau du GPU,

2. Les cartes graphiques en question sont les toutes premières cartes offrant une accélération matérielle efficace du pipeline graphique mais dont le GPU n'est pas encore programmable.

sans faire intervenir (ou seulement très peu) le processeur central, comme l'ont montré Boubekeur et Schlick [BS05]. Par conséquent appliquer un placage de déplacements par sommet devient relativement simple, comme le montre l'algorithme 3.1, et peut être effectué directement au niveau du *vertex shader*. Cette tendance est confirmée par les orientations que suivent actuellement les fabricants de processeurs graphiques et les développeurs de librairies graphiques : introduction des *geometry shaders*, des *tesselation shaders* dans les librairies graphiques DirectX 10 et 11 ainsi qu'OpenGL.

5 Placages de déplacements par pixels

Par opposition aux méthodes de placage de déplacements par sommets, celles basées sur une approche par pixels ne modifient pas directement la macrostructure et son maillage. Elles l'utilisent comme base à la déformation qui est appliquée en espace image, après projection et rastérisation de la macrostructure. Suivant cette approche, plusieurs catégories de méthodes se sont développées, en premier lieu dans le cadre d'un rendu non interactif (Pharr et Hanrahan [PH96], Heidrich et Seidel [HS98]), puis pour des rendus interactifs via des techniques basées sur des algorithmes de lancer de rayons effectués sur le GPU. D'autres techniques sont basées sur des algorithmes empruntés au rendu volumique ou aux méthodes de modélisation et de rendu basé image.

5.1 Cartes d'élévations (*Elevation maps*)

Afin d'appliquer le déplacement de la macrostructure, Dietrich [Die00] introduit les cartes d'élévations (*elevation maps*). Ces travaux adaptent une méthode de rendu basée image, développée par Schaufler [Sch98], qui permet le rendu d'imposteurs. Cette approche utilise un rendu multi-passes permettant d'émuler le déplacement de la macrostructure. Elle a l'avantage de n'introduire aucun calcul complexe à chaque passe de rendu et est entièrement accélérée par les cartes graphiques contemporaines à cette méthode.

Elle utilise des textures d'élévations qui sont de simples textures couleurs RGBα où le canal α stocke l'élévation de la surface. En d'autre termes, le champ de hauteurs est intégré à la texture couleurs via le canal α. Comme le schématise la figure 3.5, le déplacement est appliqué en extrudant de la macrostructure la boîte englobant le relief de la mésostructure. Cette boîte est rendue via plusieurs tranches, chacune associée à un intervalle d'élévations du champ de hauteurs. Le mécanisme de composition des couleurs (*blending*), permettant de gérer les surfaces transparentes qui utilise le canal α de la texture,

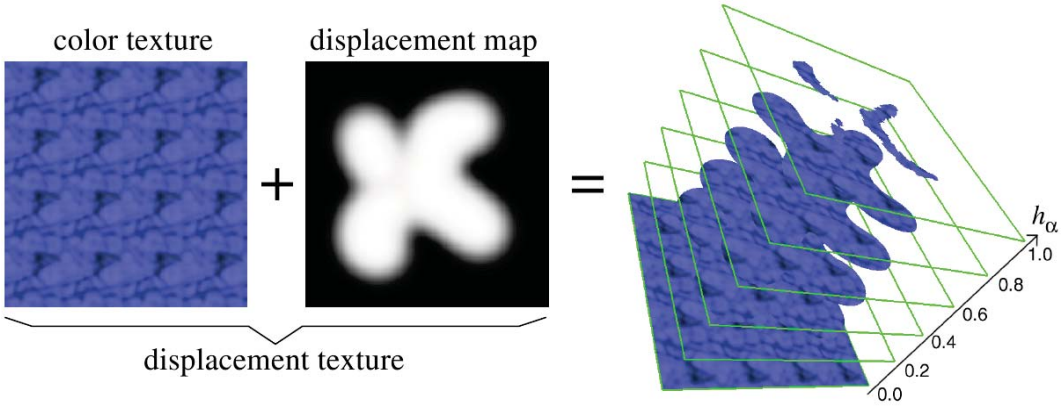


Figure 3.5 – Les *elevation maps* de Dietrich [Die00] combinent textures couleurs et champs de hauteurs. Le rendu s'effectue par tranches, dont chacune correspond à un intervalle d'élévations. Le test de transparence est activé et se base sur les élévations. Extrait de [Die00].

est activé pour rendre les différentes tranches. L'utilisation du canal α permet d'utiliser les fonctionnalités classiques de la carte graphique pour la sélection de la partie d'élévation à rendre à chaque tranche et ainsi de bénéficier de son accélération.

Suivant l'angle d'observation de l'objet, les différentes tranches peuvent devenir visibles, ce qui tend à dégrader fortement le rendu. Pour éviter ce problème, il faut adapter, en fonction de l'angle de vue, l'orientation du découpage de la boîte englobant la surface déplacée. Sur ce constat, Kautz et Seidel [KS01] proposent une méthode pour recalculer ce découpage et pour régénérer la carte de déplacements. Ils utilisent une approche développée pour le rendu de textures volumiques par Meyer et Neyret [MN98].

5.2 Textures de reliefs

Le placage de texture de reliefs (*relief texture mapping*) a été introduit par Oliveira *et al.* [OBM00]. Cette technique est basée sur les travaux de McMillan [McM97] qui proposent des méthodes de modélisation et de rendu basées image (*Image Based Modeling and Rendering*, *IBMR*) et en particulier, des travaux liés à la déformation en trois dimensions d'images (*3D image warping*). Le but de ces travaux est de générer, uniquement à partir d'images d'une scène 3D, une autre image de cette scène visualisée suivant un point de vue différent. Pour cela, il faut appliquer un certain nombre de transformations et, en particulier, une projection des pixels de l'image originale vers les pixels de la nouvelle image. Cette étape de projection a été formalisée et améliorée par McMillan

Figure 3.6 – Exemple de textures de reliefs. À gauche, une texture de couleurs contenant un ombrage pré-calculé et à droite le champ de hauteurs associé, représentant la géométrie de la mésostructure. Extrait de [POC05].

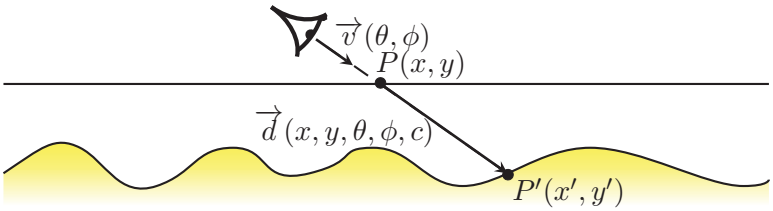


Figure 3.7 – Principe du *view-dependent displacement mapping* : la carte de déplacements représente la fonction $\overrightarrow{d}$ qui encode le déplacement à appliquer à la macrostructure pour chaque pixel en fonction du point de vue courant et de la courbure de la surface.

[McM97]. L'étape de projection est mise à profit par Oliveira *et al.* [OBM00] pour déformer l'image résultante afin d'ajouter un déplacement à la géométrie des objets. Ce déplacement est paramétré par une valeur provenant d'un champ de hauteurs.

Ce champ de hauteurs est encapsulé au sein de la structure de données employée pour ces textures de reliefs qui est, au final, assez simple : il ne s'agit que d'une texture couleurs (pouvant contenir un ombrage pré-calculé) et un champ de hauteurs représentant la géométrie des détails. Un exemple d'une telle texture est donné par la figure 3.6.

5.3 Cartes de déplacements dépendants du point de vue

Dans le cas du *view-dependent displacement mapping*, introduit par Wang *et al.* [WWT+03], la mésostructure est calculée selon une carte de déplace-

ments dépendante du point de vue. Cette carte encode pour chaque direction potentielle du vecteur de vue $\overrightarrow{v}(\theta, \phi)$ et pour chaque point $P(x_P, y_P)$ de la macrostructure, le déplacement $\overrightarrow{d}$ à appliquer à cette surface. Ce déplacement permet de déterminer le point P' réellement visible de la mésostructure. La figure 3.7 illustre le fonctionnement de cette méthode.

Les méthodes d'ajout de détails, comme le *parallax mapping* [KTI+01, Wel04], ne prennent généralement pas en compte la courbure de la macrostructure. Wang *et al.* proposent de paramétrer les cartes de déplacements par un coefficient qui représente la courbure c de la surface au point P. Par conséquent, pour rendre la mésostructure, il faut, pour tous les pixels de la macrostructure, calculer le point d'intersection P' en se basant sur l'équation suivante :

$$P' = P + \overrightarrow{d}(x_P, y_P, \theta, \phi, c)$$

Le problème majeur de cette approche se situe au niveau de la création et du stockage de la carte de déplacements. Une phase de pré-calculs est en effet nécessaire pour générer les cartes de déplacements. De plus, étant donné la taille que représente ces cartes, un mécanisme de compression/décompression est obligatoire : dans l'implantation proposée par Wang *et al.* [WWT+03], 64 Mo sont nécessaires pour stocker une carte de déplacements, une fois compressée. Cette carte est basée sur un champ de hauteurs de 128^2 échantillons en utilisant 32×8 directions de vue et 16 courbures. Ce mécanisme implique une dégradation de la qualité du rendu (présence d'artéfacts) et complique les accès à la carte de déplacements. De plus, à cause de l'occupation mémoire nécessaire, les cartes de déplacements ne peuvent être utilisées sur toutes la surface d'un objet, et restent donc cantonnées à une texture répétitive, nuisant au réalisme du rendu.

On peut noter que cette structure peut être exploitée plus en avant via certaines extensions visant à améliorer le rendu. En effet, comme le proposent Wang *et al.*, la carte de déplacements peut être utilisée pour calculer les auto-ombrages de la mésostructure. Cette technique reprend les bases de l'*horizon mapping* [Max88] permettant un calcul d'ombres dans le cas du *bump mapping*. Les cartes de déplacements dépendantes du point de vue permettent également de gérer la silhouette de l'objet. Une valeur spéciale de la carte de déplacements permet d'annuler le traitement d'un pixel s'il ne correspond à aucun point de la mésostructure.

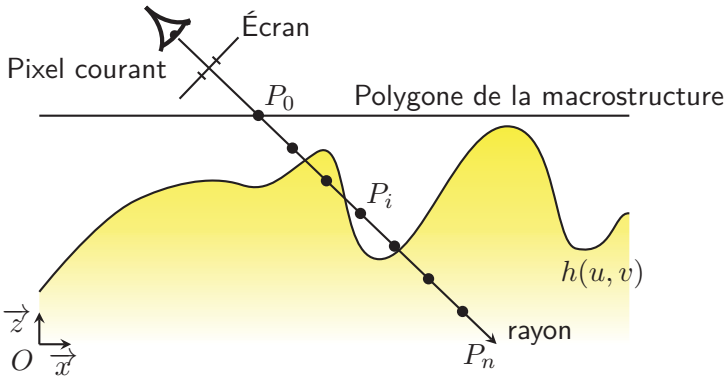


Figure 3.8 – Principe, en coupe, de fonctionnement du placage de reliefs : pour chaque pixel correspondant à un point de la macrostructure, un rayon est lancé puis est parcouru, suivant un nombre fixe de pas P_i, pour y trouver l'intersection avec la mésostructure.

6 Placages de déplacements par lancer de rayons par pixels

Les méthodes de placage de reliefs, également connues sous le nom de *relief mapping*, fonctionnent sur un principe similaire aux cartes de déplacement dépendantes du points de vue. La différence réside dans la méthode permettant d'évaluer le déplacement à appliquer à la macrostructure. Dans le cas des cartes de déplacements dépendantes du point de vue, celui-ci est pré-calculé tandis que dans le cas du placage de relief, il est calculé directement au moment du rendu. Pour cela, ces méthodes utilisent un algorithme de lancer de rayons qui recherche l'intersection entre un rayon, issue du pixel courant et suivant la direction de vue, et la mésostructure. Cela permet de calculer le déplacement à appliquer à chaque pixel de la macrostructure afin d'obtenir sa position réelle sur la mésostructure. Celle-ci étant représentée par un champ de hauteurs, la problématique de ces méthodes se ramène à déterminer, pour chaque pixel, l'intersection entre un rayon et un champ de hauteurs. On rejoint ici les problématiques rencontrées par les méthodes de visualisation de terrains basées sur des algorithmes de lancer de rayons qui sont décrites au chapitre 6.

Afin d'exécuter le lancer de rayons, la macrostructure sert de base au rendu en englobant généralement la mésostructure. Pour chaque fragment issu de la rastérisation de la macrostructure, un rayon est lancé, comme l'illustre la figure 3.8. Le rayon lancé part de l'observateur et passe par le pixel courant. Ce rayon est échantillonné, le plus souvent régulièrement par un nombre fixe [3] de points. L'utilisation comme point de départ au lancer de rayons des pixels

3. Le choix de fixer le nombre de pas est une contrainte liée aux processeurs graphiques qui présentent une baisse sensible de performances lorsqu'une boucle ne comprend pas un nombre d'itérations fixe connu à la compilation du programme de shaders.

formant la macrostructure permet de définir un point de départ proche de la mésostructure, ce qui minimise le nombre de pas à effectuer lors de la marche sur le rayon. Il faut noter que les calculs liés au parcours du rayon sont effectués en espace texture et non pas en espace objet, comme cela est généralement le cas avec des lancers de rayons classiques. Ce choix permet de simplifier les algorithmes de parcours de rayon employés, et surtout les tests d'intersections avec le champ de hauteurs. Ce type d'approche se rapproche du lancer de rayon virtuel, proposé par Dischler [Dis98], mais effectué sur le processeur central. Toutefois, effectuer le parcours des rayons en espace texture pose des problèmes assez complexes au niveau des bordures qui peuvent devenir visibles, notamment sur les silhouettes des objets.

C'est au niveau du parcours du rayon que se situent les principales différences entre les techniques de placage de reliefs. En effet, plusieurs stratégies de recherches d'intersections sont possibles, mais se basent toutes sur un parcours itératif du rayon, *ray marching* tel que le présente Levoy [Lev90]. Parmi les stratégies de recherches d'intersections, on peut citer la recherche linéaire, le recherche binaire, la recherche par la méthode de la sécante. Toutes ces méthodes ont été développées en vue d'être exécutées directement par le processeur graphique.

Le but de l'algorithme de lancer de rayons est de trouver l'intervalle dans lequel le rayon intersecte le champ de hauteurs (en d'autres termes, passe sous le champ de hauteurs, *cf.* figure 3.8). Pour cela, chaque point $P_i(x_{P_i}, y_{P_i}, z_{P_i})$, échantillonné le long du rayon, peut être exprimé dans le repère tangent au polygone de la macrostructure auquel le fragment courant appartient. Dans cet espace, on peut facilement tester si le point échantillonné se trouve au-dessus ou au-dessous de la surface de la mésostructure, définie par le champ de hauteurs. Il suffit de tester simplement si $z_{P_i} < h(x_{P_i}, y_{P_i})$, où la fonction h représente le champ de hauteurs.

Ce test d'intersections n'est valide que dans le cas où l'espace utilisé pour effectuer la progression le long du rayon est linéaire (*i.e.* la macrostructure forme localement un plan). Dans le cas contraire, le test d'intersections ne peut être effectué aussi simplement.

6.1 Recherche d'intersections entre rayons et champ de hauteurs

Comme indiqué précédemment, différentes stratégies existent pour rechercher l'intersection le long du rayon. De ces différentes stratégies dépend la précision mais également la vitesse de rendu de la méthode de placage de relief. Leur importance est donc capitale.

Entrées : $N \leftarrow$ nombre de d'échantillons ;
$E \leftarrow$ point d'entrée du rayon (P_0) ;
$S \leftarrow$ point de sortie du rayon (P_n) ;
Sorties : I : point d'intersection entre le rayon et le champ de hauteurs;

Calcul du pas entre chaque échantillon
$\overrightarrow{d} \leftarrow \frac{\overrightarrow{ES}}{N}$;
$P \leftarrow E$;
$intersectionTrouvée \leftarrow$ `faux` ;
Parcours linéaire du rayon
pour $i \leftarrow 0$ **à** N **faire**
  si $z_P < \mathtt{h}(x_P, y_P)$ **alors**
    $I \leftarrow P$;
    $intersectionTrouvée \leftarrow$ `vrai` ;
  sinon
    $P \leftarrow P + \overrightarrow{d}$;
  fin
fin

Algorithme 3.2 – Algorithme de recherche linéaire d'intersection avec un champ de hauteurs : les rayons sont parcouru linéairement jusqu'à ce que le rayon passe sous la surface du champ de hauteurs.

Recherche linéaire La *recherche linéaire* est l'algorithme de recherche le plus simple. Les points sont échantillonnés régulièrement tout au long du rayon, entre le point d'entrée P_0 dans un volume englobant la mésostructure (*cf.* figure 3.8) et le point P_n de sortie du rayon hors de ce même volume englobant. Le parcours du rayon s'effectue linéairement, comme l'indique l'algorithme 3.2.

Cet algorithme appelle à quelques commentaires. Cette méthode de recherche autorise, sous réserve d'un échantillonnage suffisamment dense, la détection d'une intersection entre le rayon et la surface ou, plus exactement, permet de trouver l'intervalle dans lequel se trouve la racine $(z_{P_i} - h(x_{P_i}, y_{P_i}) = 0)$. Dans le cas où il n'y en a pas, cette recherche offre la possibilité d'annuler le traitement du pixel courant. Cela permet de gérer plus fidèlement la silhouette de l'objet. La recherche linéaire est cependant très dépendante de la fréquence d'échantillonnage du rayon, c'est-à-dire de la distance séparant chaque point échantillonné le long du rayon : si celle-ci est trop grande par rapport aux reliefs de la mésostructure, des intersections peuvent être manquées. L'intersection retournée n'est, dans ce cas, pas nécessairement la première le long du rayon. De plus, la position du point d'intersection n'est qu'approximée et la qualité de cette approximation dépend de la fréquence d'échantillonnage. Pour affiner le résultat, elle est généralement combinée avec une autre méthode de recherche

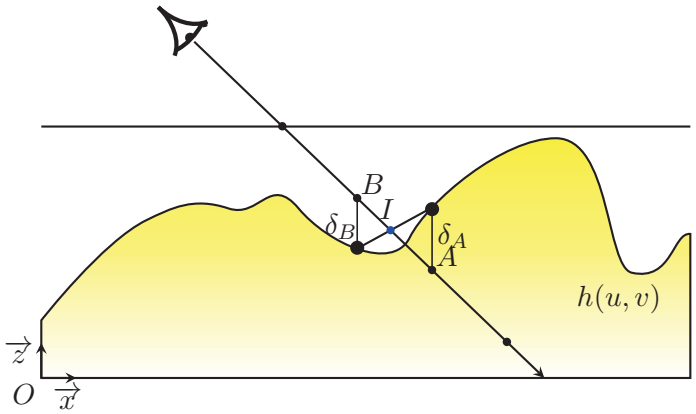


Figure 3.9 – Recherche d'intersections par la méthode de la sécante : l'intersection est approximée en calculant sa position à partir du segment AB l'encadrant et de l'altitude de ces deux points.

d'intersections : binaire pour Policarpo *et al.* [POC05], par une méthode de la sécante pour Tatarchuk [Tat06a].

Méthode de la sécante Basé sur le même principe qu'une recherche linéaire, la recherche par la *méthode de la sécante* utilise cependant un test d'intersections différent. Elle utilise, pour cela, les deux échantillons qui encadrent le point d'intersection, comme le décrivent la figure 3.9 et l'algorithme 3.3. Cette méthode pose l'hypothèse que la surface du champ de hauteurs est plane entre les points A et B qui encadrent le point d'intersection. Elle en détermine, à partir des différences d'altitudes δ_A et δ_B, la position du point I d'intersection entre le rayon et le champ de hauteurs. Le point I peut ainsi être calculé à partir de cette équation :

$$I = B + \frac{\delta_A}{\delta_A - \delta_B} \cdot \overrightarrow{BA}$$

Pour affiner la précision du résultat, ce calcul peut être répété plusieurs fois, comme le montre l'algorithme 3.3. Cette méthode de recherche d'intersections est, en particulier, utilisée par Yerex et Jagersand [YJ04]. Tatarchuk [Tat06a] utilise également cette méthode, mais uniquement avec une seule étape, pour affiner le résultat de sa recherche linéaire.

Recherche dichotomique La *recherche dichotomique*, connue également sous le nom de *recherche binaire*, fonctionne suivant le principe décrit par l'algorithme 3.4. Cette recherche à l'avantage de converger plus rapidement que la recherche linéaire. Elle est également plus précise que la recherche linéaire, à nombre de pas égal. Son principal défaut réside dans le fait que l'intersection retournée n'est pas nécessairement la première le long de rayon. De plus

Entrées : $N \leftarrow$ nombre d'itérations de recherche ;
$A \leftarrow$ premier point du rayon sous la surface ;
$B \leftarrow$ dernier point du rayon avant la surface ;
Sorties : I : point d'intersection entre le rayon et le champ de hauteurs ;

pour $i \leftarrow 0$ **à** N **faire**
  $\delta_A = h(x_a, y_a) - z_A$;
  $\delta_B = h(x_b, y_b) - z_B$;
  $I \leftarrow B + \frac{\delta_A}{\delta_A - \delta_B} \cdot \overrightarrow{BA}$;
  si $z_I \leqslant h(x_I, y_I)$ **alors**
    $A \leftarrow I$;
  sinon
    $B \leftarrow I$;
  fin
fin

Algorithme 3.3 – Algorithme de recherche d'intersections par la méthode de la sécante : la position du point d'intersection est affinée en réduisant plusieurs fois le segment le contenant et défini par deux points d'échantillonnage du rayon.

Entrées : $N \leftarrow$ nombre de pas de recherche ;
$E \leftarrow$ point d'entrée du rayon (P_0) ;
$S \leftarrow$ point de sortie du rayon (P_n) ;
Sorties : I : point d'intersection entre le rayon et le champ de hauteurs ;

Initialisation de l'espace de recherche
$P_{max} \leftarrow E$;
$P_{min} \leftarrow S$;
pour $i \leftarrow 0$ **à** N **faire**
  $P \leftarrow P_{max} + \frac{\overrightarrow{P_{max}P_{min}}}{2}$;
  si $z_P \leqslant h(x_P, y_P)$ **alors**
    $P_{min} \leftarrow P$;
  sinon
    $P_{max} \leftarrow P$;
  fin
fin

Algorithme 3.4 – Algorithme de recherche binaire d'intersections avec un champ de hauteurs. Son fonctionnement est basée sur le principe de la recherche dichotomique qui, à chaque pas, échantillonne le rayon au milieu du segment contenant l'intersection.

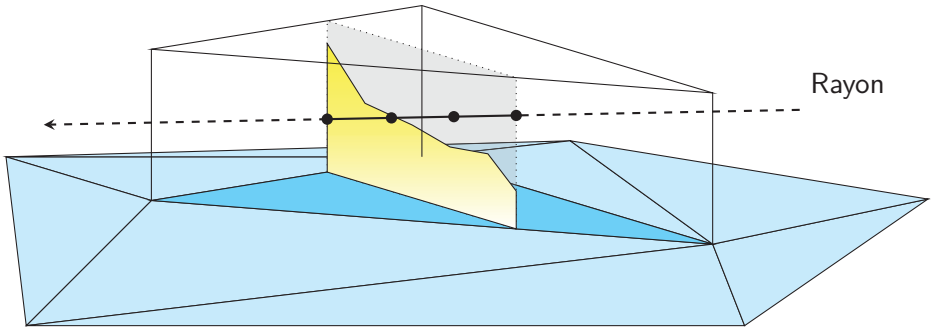


Figure 3.10 – Parcours de rayon à travers un prisme développé par [HEGD04]. Le prisme est extrudé à partir des polygones formant la macrostructure (en bleu).

cet algorithme ne permet pas de décider s'il existe réellement une intersection entre le rayon et la surface du champ de hauteurs. Par conséquent, cette recherche est principalement utilisée pour affiner le résultat de la recherche linéaire [POC05, PO06].

6.2 *Relief mapping*

Comme mentionné lors de la description des méthodes de recherche d'intersections, différentes techniques de placage de reliefs basées sur ces tests d'intersections entre rayon et champ de hauteurs ont été développées. La plupart des techniques combinent différents algorithmes de recherche pour optimiser la précision et la vitesse d'exécution.

Hirche *et al.* [HEGD04] proposent une méthode utilisant uniquement une recherche linéaire. Son originalité réside dans la forme géométrique de base utilisée pour exécuter le lancer de rayons. En effet, chaque triangle, formant le maillage de la macrostructure, est extrudé afin de former un prisme, comme l'illustre la figure 3.10. Ce prisme est subdivisé en trois tétraèdres qui sont rendus via l'algorithme des tétraèdres projetés de Shirley et Tuchman [ST90]. Pour chaque pixel couvert par l'un de ces tétraèdres, un rayon est parcouru. L'intersection sur ce rayon est simplement déterminée, via une recherche linéaire, en comparant l'altitude du point échantillonné avec l'altitude stockée dans la carte de hauteurs. Bien que pouvant gérer les silhouettes de l'objet, cet algorithme reste complexe à implanter sur GPU, notamment à cause du prisme utilisé pour le lancer de rayons.

Le *relief mapping* [POC05] et le *parallax occlusion mapping* [Tat06a], sont deux techniques très proches. Le *relief mapping* combine une recherche linéaire à une recherche binaire pour calculer l'intersection entre le rayon et

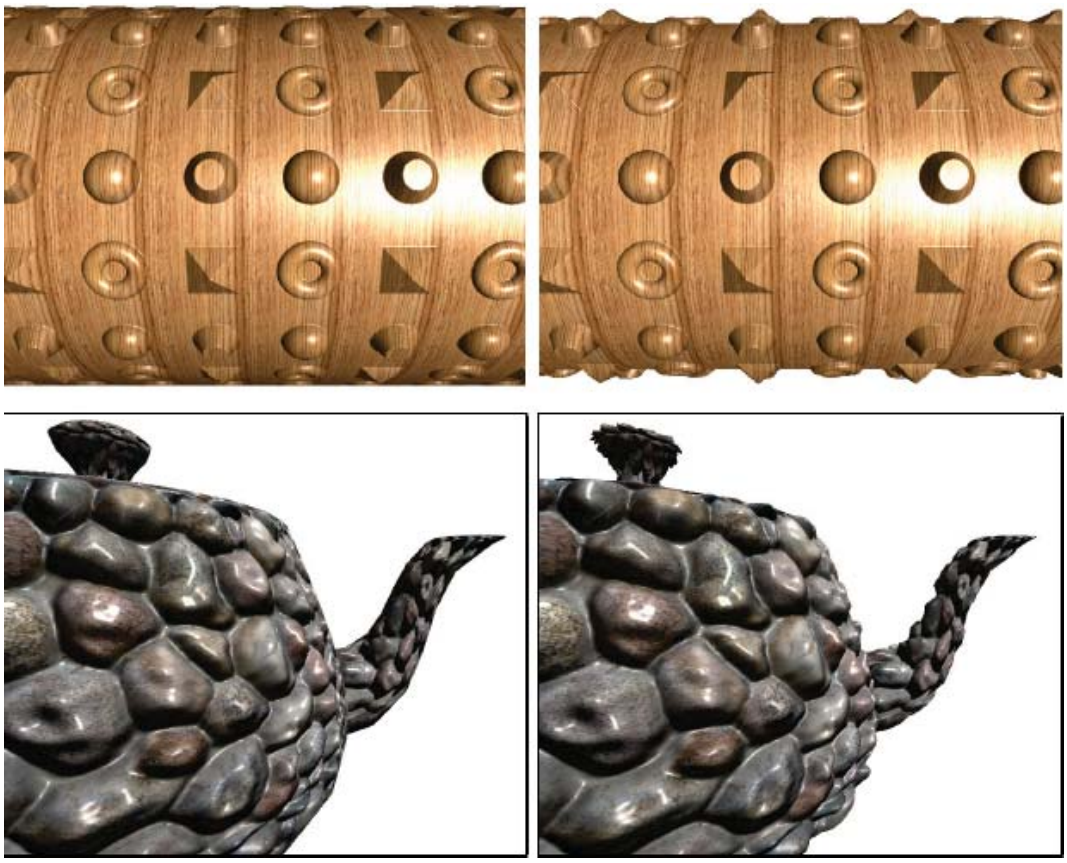


Figure 3.11 – Gestion de du relief de la silhouette. À gauche *relief mapping* sans gestion des silhouettes [POC05]. À droite avec gestion des silhouettes [OP05]. Extrait de [OP05].

le champ de hauteurs. La recherche binaire est exécutée sur l'intervalle borné par le premier point sous la surface du champ de hauteurs et le dernier point au dessus.

Comme nous l'avons mentionné précédemment, la recherche d'intersections entre le rayon et le champ de hauteurs s'effectuant en espace de texture, il peut y avoir des problèmes en bordure des polygones. Ainsi, dans la version initiale de ces travaux [POC05], la silhouette de l'objet n'est pas correctement gérée, comme l'illustre la figure 3.11. Pour remédier à ce problème Oliveira et Policarpo [OP05] proposent de recourir à des quadriques pour encoder la courbure de la surface de l'objet. Une quadrique est associée à chaque sommet de la macrostructure. Cette information est utilisée pendant le lancer de rayons pour déformer localement la surface de la macrostructure lors du calcul d'intersection entre le rayon et le champ de hauteurs. Cela permet ainsi de gérer correctement la recherche d'intersections dans le cas des polygones formant la silhouette de l'objet.

Le *parallax occlusion mapping* [BT04, Tat06a, Tat06b] reprend le même principe. D'après Tatarchuk [Tat06a], obtenir le point d'intersection directement en interrogeant un texel d'une texture pose un problème de précision : la valeur retournée est codée sur 8 bits seulement. Par conséquent, des artéfacts sont visibles si les reliefs sont trop importants. Pour minimiser ces artéfacts, Policarpo *et al.* [POC05] proposent d'appliquer un biais sur les valeurs de profondeurs utilisées. Cette opération a pour conséquences d'écraser et d'aplatir les reliefs, surtout dans le lointain. Pour éviter ce problème, Tatarchuk [Tat06a] propose d'utiliser un test d'intersection par la méthode de la sécante. Ce couplage permet de tirer parti d'un calcul du point d'intersection s'effectuant sur 32 bits et évite ainsi d'avoir recours à l'application d'un biais sur les altitudes du champ de hauteurs qui écrase les reliefs.

7 Techniques d'optimisations de la recherche d'intersections

Pour des questions de performances, notamment à cause de l'efficacité des branchements conditionnels sur GPU, toutes ces techniques de placage de reliefs, basées sur des algorithmes de lancer de rayons [HEGD04, POC05, BT04, Tat06a] échantillonnent le rayon avec un nombre de points choisi de façon arbitraire. Ce nombre d'échantillons est fixé, soit par le développeur, soit par l'artiste ayant recours au shader et, est, par conséquent, indépendant des champs de hauteurs utilisés. Ainsi, si le pas d'échantillonnage est choisi trop grand, il est possible que des intersections entre les rayons et la surface du champ de hauteurs soient manquées, ce qui se traduit par des artéfacts visibles lors des changements de point de vue. Si le pas est trop petit, le parcours du rayon devient pénalisant en terme de vitesse. Afin de minimiser ces problèmes et d'accélérer la recherche d'intersections, des techniques d'optimisation dédiées à la recherche d'intersections ont été proposées.

Dans le cas d'une recherche linéaire, l'utilisation d'un pas d'échantillonnage fixe pose des problèmes de performances. En effet, avant de trouver l'intersection, il faut fréquemment traverser une large zone au-dessus de la mésostructure. Ce parcours introduit des tests d'intersection supplémentaire qui se révèlent coûteux [4]. De plus, pour garantir une position exacte du point d'intersection, il faut définir un échantillonnage suffisamment dense le long du rayon et qui soit dépendant de sa trajectoire sur le plan de base du champ de hauteurs. Cela pose des problèmes au niveau de l'implantation sur GPU. En effet, dans le cas où l'échantillonnage dépendant de la direction et de l'orientation

4. Un test d'intersection nécessite au minimum un accès à la texture représentant le champ de hauteurs. Ces accès mémoire peuvent causer des latences lors de l'exécution de lancer de rayons sur le GPU.

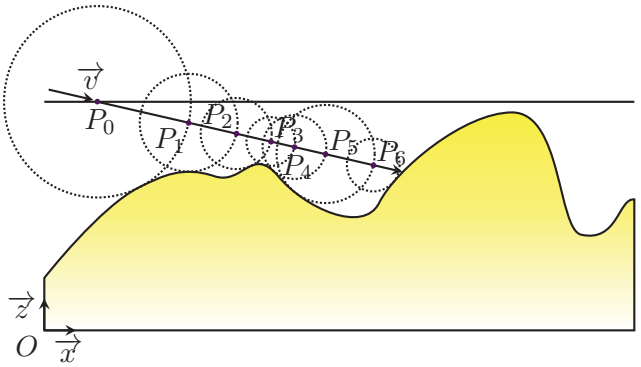


Figure 3.12 – Parcours de rayon en utilisant une fonction de distances décrite par [Don05]. À chaque pas, la carte de distances est interrogée pour déterminer la taille du pas suivant. La carte garantit qu'aucune intersection ne sera manquée.

du rayon, le nombre de pas à effectuer sur le rayon ne peut pas être connu à l'avance mais est dépendant de l'orientation du rayon. Cela implique d'utiliser une boucle dont le nombre de pas n'est pas connu au moment de la compilation, ce qui pose de sérieux problèmes de performances. C'est pourquoi des techniques d'optimisation du placage de textures de reliefs ont été développée pour proposer des solutions pour accélérer la recherche d'intersection et également sa précision.

De telles techniques ont également été proposées dans le cadre la visualisation de terrains, comme celles décrites plus tard, à la section 4 du chapitre 6. Cependant, leur application au GPU est difficilement envisageable étant données les contraintes qu'impose ce type de processeurs. Des techniques ont également été développées pour implanter des algorithmes de lancer de rayons sur le processeur graphique (on peut citer les travaux de Purcell *et al.* [PBMH02]). Cependant ces travaux s'attachent à étudier le cas général et ne sont pas optimales pour le cas, très spécifique, du lancer de rayons sur champs de hauteurs.

Des techniques d'optimisation spécifiques au placage de textures de reliefs ont donc été développées. On peut dégager deux approches distinctes. La première approche consiste à définir une carte de distances permettant d'adapter le pas d'échantillonnage du rayon en fonction du relief du champ de hauteurs (sous-section 7.1). La seconde approche se base sur l'utilisation d'une structure hiérarchique (sous-section 7.2), dans l'esprit du mipmapping [Wil83].

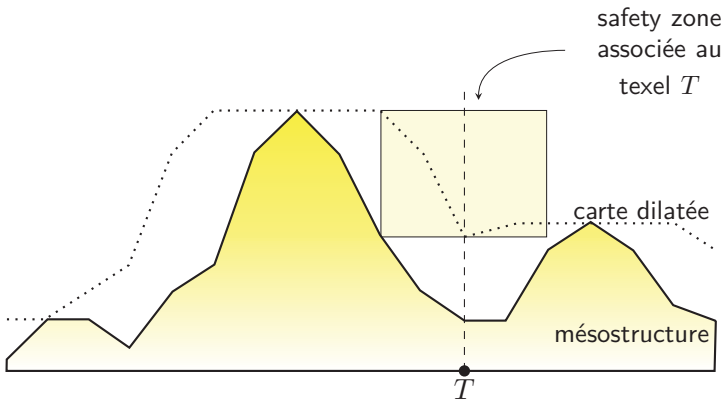


Figure 3.13 – *Safety zone* associée à un texel donné et une dilatation donnée d'une carte d'élévations. Cette zone, calculée pour chaque élévation du champ de hauteurs, garantit qu'aucune intersection avec le champ de hauteurs n'y est présente.

7.1 Méthodes d'optimisation par cartes de distances

Donnelly [Don05] présente une technique d'accélération basée sur une fonction de distances. Cette fonction de distances permet de représenter la distance minimale entre un point P et la mésostructure de la surface $\mathcal{S}$ de l'objet :

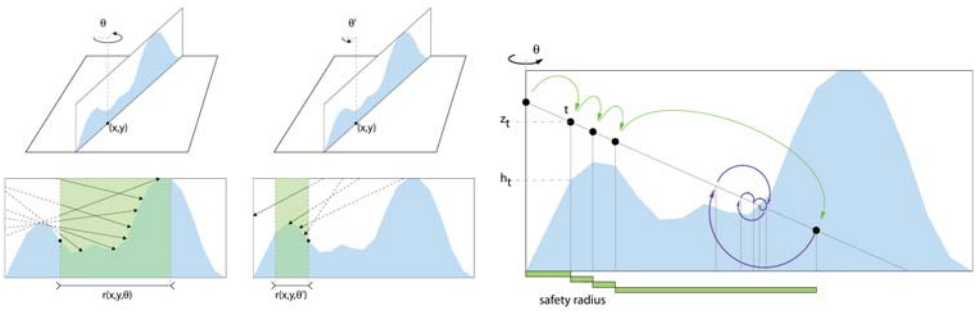
$$dist(P, \mathcal{S}) = \min\{d(P, Q), Q \in \mathcal{S}\}$$

La fonction de distances est encodée dans une texture à trois dimensions. Elle est calculée par un algorithme basé sur une transformée en distances [Dan80]. Le parcours du rayon s'effectue, comme l'illustre la figure 3.12, en avançant pas à pas, mais la taille de ces pas est dépendante de la valeur de la fonction de distances au point d'échantillonnage P_i courant du rayon :

$$P_{i+1} = P_i + dist(P_i, \mathcal{S}) \cdot \overrightarrow{v}$$

Le recours à une texture en trois dimensions augmente l'occupation mémoire du placage de reliefs et son utilisation devient inefficace lorsque celle ci devient trop volumineuse[5]. Cette carte de distances peut-être ramenée à une carte à deux dimensions comme le montrent Kolb et Rezk-Salama [KRS05] ainsi que Baboud et Décoret [BD06]. De façon analogue à Donnelly, ces deux méthodes définissent une zone de sécurité permettant de moduler la taille du pas d'échantillonnage du rayon.

5. Lors de l'utilisation de textures en trois dimensions trop volumineuses, on peut noter des problèmes de performances lors des accès à la texture, des problèmes liés à la mise en mémoire cache, etc.



(a) Région définie par le *safety radius* où, pour un texel et une orientation de rayon donnés, il n'y a qu'une seule intersection.

(b) Recherche binaire en utilisant la valeur du *safety radius*.

Figure 3.14 – Principe de fonctionnement du *safety radius*. Extrait de [BD06].

La technique de Kolb et Rezk-Salama [KRS05] s'inscrit dans la continuité des travaux de Hirche *et al.* [HEGD04]. Ils proposent de calculer une zone de sécurité permettant d'accélérer le parcours des espaces vides au dessus du champ de hauteurs. Cette zone de sécurité est calculée en utilisant un mécanisme de dilatation et d'érosion, emprunté au domaine du traitement d'images. La figure 3.13 présente la zone de sécurité associée à un texel du champ de hauteurs. Ces deux mécanismes sont appliqués sur le champ de hauteurs et permettent de générer deux cartes utilisées pour calculer la segment du rayon qui encadre l'intersection avec la surface du champ de hauteurs. Ce segment est utilisé pour exécuter une recherche binaire afin de déterminer précisément la position du point d'intersection entre le rayon et la mésostructure de la surface.

Baboud et Décoret [BD06] introduisent la notion de rayon de sûreté (*safety radius*). Ce rayon est une fonction $r(t, \theta)$ définie pour un texel t de la carte de hauteurs et une orientation θ du rayon[6]. Pour un texel donné et suivant une direction de rayon donné, cette fonction représente la région dans laquelle il y a, au plus, une intersection avec le champ de hauteurs (*cf.* figure 3.14a). La fonction représentant le *safety radius* est échantillonnée puis encodée dans une carte à deux dimensions. Une valeur de *safety radius* est calculée pour chaque élévation de la carte de hauteurs. Ce calcul est effectué en échantillonnant densément les différentes directions que peuvent prendre les différents rayons passant par le texel courant de la carte de hauteurs. Pour chacune de ces directions, une distance de sécurité est calculée et seule la valeur maximale de ces distances est conservée.

En utilisant la propriété définissant le *safety radius*, la recherche d'in-

6. On peut voir cette orientation comme l'azimuth du rayon dans l'espace tangent.

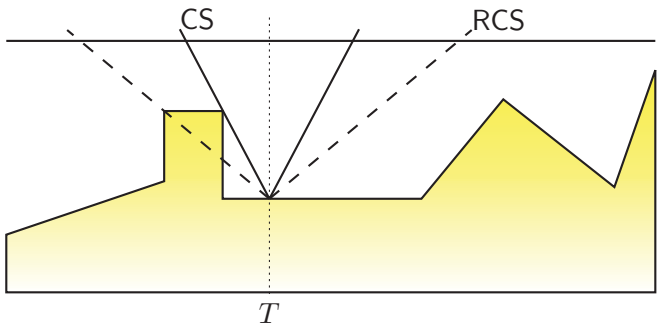


Figure 3.15 – *Cone step mapping* (CS) et *relaxed cone step mapping* (RCS) : le CS garantit qu'il n'y a aucune intersection à l'intérieur tandis que le le RCS garantit qu'il y a au plus une intersection dans le cône. Dans les deux cas, les cônes correspondent au même texel.

tersection peut s'effectuer suivant le même principe que le *relief mapping* de Policarpo *et al.* [POC05] : une recherche linéaire pour passer une première fois sous la surface du champ de hauteurs puis une recherche binaire pour affiner la position du point d'intersection. Le pas d'échantillonnage est donné par le *safety radius* (*cf.* figure 3.14b). Il est appliqué de la même manière que dans le cadre de la méthode de Donnelly [Don05].

Une autre technique d'optimisation, le *cone step mapping* a été initialement proposée par Dummer [Dum06]. Cette méthode a été reprise et étendue, sous le nom de *relaxed cone stepping*, par Policarpo et Oliveira [PO07]. La technique initiale de Dummer définit, pour chaque élévation de la carte de hauteurs, un cône dont l'ouverture est choisie de sorte qu'il n'y ait aucune intersection entre le cône et la surface du champ de hauteurs. La figure 3.15 illustre ce mécanisme. Le parcours du rayon s'effectue en passant d'un bord du cône à l'autre suivant la direction du rayon jusqu'à intersecter la surface du champ de hauteurs. Dans leur extension, Policarpo et Oliveira relâchent la contrainte qui définit les cônes pour en augmenter l'ouverture. La contrainte définissant les cônes devient la suivante : un rayon traversant un cône perce la surface du champ de hauteurs au maximum une fois (*cf.* figure 3.15). L'algorithme de parcours du rayon s'en trouve par conséquent modifié. Il débute via la même recherche que Dummer [Dum06], jusqu'à passer sous la surface du champ de hauteurs. Cette position ainsi que la position précédente sur le rayon forme un intervalle sur lequel est exécutée une recherche binaire.

On peut noter que ces différentes méthodes ont en commun une phase de pré-calculs pouvant être assez longue, ce qui ne permet pas de les utiliser sur des données qui sont dynamiques. On peut cependant noter que le gain de performances par rapport au placage de reliefs non optimisé [POC05, Tat06a]

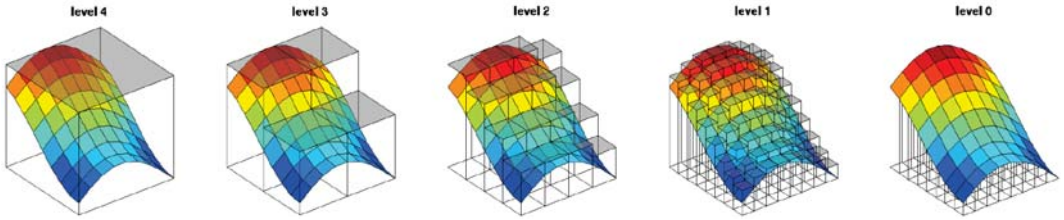


Figure 3.16 – *Maximum mipmaps* : pour chaque niveau de détails est définit une altitude basé sur l'altitude maximum des quatre élévations correspondantes du niveau de détails inférieur. Extrait de [TIS08].

est tout de même assez sensible pour un surcoût mémoire assez raisonnable[7] (sauf dans le cas de la carte à trois dimensions nécessaire à Donnelly [Don05] pour encoder la carte de distances).

7.2 Méthodes d'optimisation par structures hiérarchiques

Une autre approche, permettant d'accélérer et d'optimiser le placage de reliefs, consiste à utiliser une structure hiérarchique inspirée des *mipmaps*. Son implantation pour l'adapter aux cartes de hauteurs s'apparente au principe des quad-trees [Sam84, Sam90]. Pour cela, Oh *et al.* [OKL06] introduisent la notion de placage pyramidal de déplacements (*pyramidal displacement mapping*). Cette approche est présentée, sous une autre forme et de manière plus complète, par Tevs *et al.* [TIS08] en introduisant le principe des *maximum mipmaps*. Parallèlement à Oh *et al.*, Schroders et Gulik [SG06] introduisent la technique du *quadtree relief mapping*, fonctionnant sur le même principe.

Ces structures de données sont toutes basées sur les mêmes principes : les quadtrees et le mipmapping. Elles consistent à créer une pyramide de champs de hauteurs dont la résolution est divisée par deux à chaque niveau, et ce jusqu'à une carte ne comportant qu'une seule valeur (*cf.* figure 3.16). Le niveau 0 de la pyramide est formé par le champ de hauteurs original. L'élévation d'une cellule (d'un texel) du niveau l est donnée par le maximum des élévations des quatre cellules correspondantes du champ de hauteurs de niveau $l-1$.

Tevs *et al.* utilisent l'algorithme 3.5 pour parcourir le rayon. Les calculs d'intersection sont effectués entre le rayon et les plans formant une cellule de la hiérarchie au niveau considéré ou avec le champ de hauteurs dans le cas du niveau de mipmap le plus fin[8]. Contrairement à Oh *et al.*, le parcours

7. Les méthodes développées par Baboud et Décoret [BD06] et Policarpo et Oliveira [PO07] ne nécessitent qu'une seule valeur supplémentaire par élévation de la carte de détails.

8. Dans ce cas l'intersection avec le champ de hauteurs est calculée en reconstruisant

Entrées : $P \leftarrow$ point d'échantillonnage courant ;
$\overrightarrow{v} \leftarrow$ vecteur de vue ;
Sorties : I : point d'intersection;

Niveau de détails
$l \leftarrow N$;
tant que P *est au dessus du champs de hauteurs* **faire**
  $h \leftarrow$ `altitudeMaximum`$(P, \overrightarrow{v}, l)$;
  si $l = 0 \wedge z_P \leqslant h$ **alors**
    $I \leftarrow$ `calculIntersectionAvecChamp`$(P, \overrightarrow{v})$;
  sinon
    $I \leftarrow$ `calculIntersectionAvecPlansEnglobants`$(P, \overrightarrow{v}, l)$;
  fin
  si $z_P \geqslant h \vee l = 0$ **alors**
    $P \leftarrow I$;
    $l \leftarrow$ `metAJourNiveau`(l, P, h);
  sinon
    $l \leftarrow l - 1$;
  fin
fin

Algorithme 3.5 – Parcours de rayon dans le cas des *maximum mipmaps* [TIS08].

s'effectue directement en utilisant les échantillons. Cela permet d'utiliser des coordonnées entières pour la position des échantillons et ainsi d'éviter des artéfacts au moment du rendu. La méthode de Tevs *et al.* offre, en outre, la possibilité de remonter dans les niveaux de détails lorsque nécessaire[9] (méthode `metAJourNiveau` de l'algorithme 3.5). Cette approche hiérarchique permet également d'offrir un mécanisme de niveaux de détails pour dégrader la qualité de rendu lorsqu'un plus faible niveau de précision est suffisant (zones lointaines, par exemple).

Au final, ce type de structures de données permet un gain de performances important par rapport aux techniques de placage de reliefs non optimisées, telles que décrites par Policarpo *et al.* [POC05] ou Tatarchuk [Tat06a]. Ces performances sont obtenues tout en maintenant une structure de données accélératrice facile et rapide à calculer, contrairement aux méthodes basées sur une carte de distances [Don05, BD06, PO07] qui nécessitent un échantillonnage dense des différents paramètres (direction de rayon, ...).

un patch bilinéaire à partir des quatre texels adjacents.

9. La stratégie employée par Tevs *et al.* est assez simple et se limite à incrémenter le niveau de détail courant.

Au niveau des performances de recherche d'intersections, et comme le montre Schroders et Guli [SG06], la complexité de ces algorithmes d'optimisation basée sur une structure hiérarchique se situe théoriquement entre $O(\log(p))$ et $O(\sqrt{p})$ mais dans la pratique, une complexité en $O(\log(p))$ est plus fréquemment observée.

8 Acquisition des détails

Les méthodes, décrites jusqu'à présent, supposent que les détails sont fournis sous la forme d'un champ de hauteurs. Toutefois, il convient de se demander d'où proviennent ces données. Bien que ces données soient très souvent issues du travail de graphistes, elles peuvent également être extraites de données réelles ou encore acquises via des outils de numérisation reposant sur des dispositifs de numérisation en trois dimensions. Nous présentons donc brièvement, dans cette section, différents travaux, dont le but est de fournir des champs de hauteurs qui représentent la mésostructure de la surface d'un objet. On peut noter que ces travaux sont très souvent liés à ceux destinés à générer des fonctions représentant les propriétés lumineuses de différents matériaux (*BRDF*, *BTF*, ...).

L'utilisation de dispositifs dédiés à la numérisation en trois dimensions d'objets réels, peut naturellement être employée afin d'acquérir ces mésostructures. Cependant, bon nombre de travaux se sont attaqués à ces aspects en tentant d'extraire ces informations à partir de multiples images, en se basant sur le principes de vue stéréographiques [SCD+06]. Cette technique consiste à extraire une information géométrique à partir de plusieurs vues d'un même objets, prises pour différentes positions de caméra. De façon similaire, une autre approche consiste à prendre plusieurs vues d'un même objet, mais cette fois en ne faisant varier que la position de la source lumineuse, la caméra restant fixe. Ce type d'approche est notamment employé pour extraire des champs de hauteurs, afin d'en dériver des cartes de normales. Elle est ensuite utilisée afin de rendre la surface mésostructure de l'objet numérisé via un algorithme de *bump mapping*. De telles travaux ont été menés, en particulier, par Rushmeier *et al.* [RTG97].

La nécessité d'extraire le relief de la mésostructure est également une contrainte lors de l'acquisition et de la génération de fonctions de réflectance bi-directionnelle ou de fonctions de texture bi-directionnelle. Là aussi, la reconstruction du champ de hauteurs se base sur une technique employant des captures de la géométrie, via une série d'images d'un échantillon de matériaux, prise pour différentes positions d'éclairage. On peut, à titre d'exemple, citer

les travaux menés pour la reconstruction de *BRDF* variant spatialement de Ruiters et Klein [RK09]. Ces travaux sont basés sur l'extraction d'un champ de hauteurs, permettant d'estimer les inter-réflexions lumineuses se produisant à la surface du matériaux et ainsi d'optimiser la reconstruction de la *BRDF*.

À travers les quelques méthodes présentées, on voit qu'il existe plusieurs types d'approches permettant de générer, à partir d'objets réels, des cartes d'élévations représentant le relief de la mésostructure qui forme la surface d'un objet.

9 Méthodes d'ajout de détails : un bilan

Dans ce chapitre, nous avons présenté les différentes approches développées pour ajouter des détails à la macrostructure d'un objet, à partir de données à deux dimensions et demie, afin de former la mésostructure. L'utilisation d'une carte de déplacements sous forme d'un champ de hauteurs permet d'exprimer les déplacements localement aux polygones de la macrostructure. En effet, les déplacements sont exprimés et calculés dans l'espace tangent à ces polygones. Cela nécessite cependant de disposer d'une paramétrisation de la macrostructure associant aux sommets du maillage de la macrostructure, les données de la carte de déplacements.

Pour générer la mésostructure à partir de la macrostructure, deux catégories distinctes d'approches se dégagent. La première catégorie se base sur une simple simulation de la mésostructure sans ajout réel de géométrie. Ces méthodes offrent les performances de rendu les plus élevées. Elles sont entièrement supportées et accélérées par les processeurs graphiques actuels et sont facilement mises en œuvre. Leur principal défaut réside dans les erreurs de parallaxe commises du fait de l'absence de géométrie pour représenter la mésostructure. Cela se traduit par des surfaces qui restent planes, notamment au niveau des silhouettes de l'objet où la structure grossière de la macrostructure reste fortement visible.

La seconde catégorie de méthodes ajoute réellement la géométrie de la mésostructure à la macrostructure. Il en découle une qualité visuelle plus élevée que pour les méthodes de la première catégorie. Cependant leur mise en œuvre est généralement plus compliquée et requiert des calculs plus lourds au moment du rendu et parfois durant une phase de pré-traitements. Ces calculs devraient théoriquement pénaliser les performances de rendu, l'implantation GPU de ces techniques leur permet tout de même d'atteindre des performances de rendu temps réel. Néanmoins, ces techniques ne sont pas exemptent de défauts.

Ainsi, les méthodes effectuant un placage de déplacements par sommets manipulent un maillage fin pour représenter la mésostructure. Ce maillage doit être, soit stocké, soit généré ce qui impacte les performances générales : problèmes de stockage, accès mémoire, étape de génération ou de subdivision du maillage, etc. Ces étapes de subdivision et de simplification peuvent également entraîner des déformations sur la mésostructure et par conséquent, introduire des artéfacts visuels.

Les méthodes de placage de déplacements par pixels sont également sujettes à des artéfacts de rendu. Pour les méthodes basées sur un lancer de rayons (*relief mapping* et *view-dependent displacement mapping*), ces artéfacts peuvent être réduits en augmentant la précision du lancer de rayons ce qui n'est possible, pour conserver le rendu temps réel, qu'en ayant recours à des structures accélératrices nécessitant un espace mémoire plus important et/ou des étapes de pré-calculs pouvant être assez lourdes.

Il est également utile de préciser que les différentes méthodes, présentées dans ce chapitre, ont été développées pour plaquer des détails sur des objets formés par des maillages quelconques, représentant en particulier des surfaces courbées. Par conséquent, ces différentes méthodes, et notamment les méthodes basée sur un lancer de rayons par pixels, fonctionnent en associant un champ de hauteurs à chaque polygone formant le maillage de l'objet. Cela limite généralement l'espace où s'effectuent les déplacements de la macrostructure. Dans le cas des méthodes que nous développons dans la suite de ce mémoire, le support de base se limite à des surfaces planes.

Contexte d'utilisation actuel Si le *bump mapping* a été l'une des premières méthodes d'ajout de détails proposée, elle n'en reste pas moins très utilisée de nos jours, malgré ses défauts. Cependant, la démocratisation et la montée en puissance des GPU programmables permettent d'utiliser de plus en plus fréquemment les diverses techniques de placage de reliefs basées sur un algorithme de lancer de rayons. De même que l'apparition de mécanismes (geometry shader, tesselation shaders) permettant de générer de la géométrie directement sur le GPU ouvre la voie à une utilisation performante du placage de déplacements par sommets.

Extensions possibles Ces méthodes, et plus spécialement les méthodes basées sur un traitement par pixels, autorisent un certains nombre d'extensions. La première extension concerne le calcul d'ombres. Par exemple, dans le cas du *bump mapping*, Max [Max88] a proposé un mécanisme permettant le calcul d'ombres en plus du *bump mapping*[10]. Le *relief mapping* de Policarpo et

10. Cette technique a été portée sur GPU par Sloan et Cohen [SC00]

Oliveira [POC05] ainsi que le *view-dependent displacement mapping* de Wang et al [WWT+03] introduisent également un mécanisme simple d'ombrage (ne générant que des ombres franches) qui fonctionne en relançant un rayon dans la direction de chaque source lumineuse. Dans le cas du *relief mapping* de Policarpo et Oliveira, la recherche le long de ces nouveaux rayons se limite à une recherche linéaire. En effet, une précision plus faible sur la position de ce point d'intersection est suffisante : on cherche juste à déterminer s'il existe ou non une occlusion entre le rayon lumineux et un point de la mésostructure. Ce mécanisme est amélioré par Tatarchuk [Tat06a] qui propose de générer des ombres douces. Elle propose un mécanisme qui estime, via une heuristique basée sur des méthodes traitant du problème des ombres douces [CD03, WH03], la taille de la zone du champ de hauteurs occultant la source lumineuse éclairant l'objet.

On peut également noter, dans les travaux de Policarpo *et al.* [POC05, PO06], que les auteurs proposent d'étendre cette méthode à la visualisation d'imposteurs. Pour cela, ils utilisent deux textures de reliefs : une pour la face avant de l'objet et l'autre pour la face arrière de l'objet.

Un des intérêts des méthodes qui utilisent un traitement par pixels est qu'il autorise l'application d'un mécanisme de niveau de détails avec une variabilité à l'echelle des pixels. Tatarchuk [Tat06a] donne ainsi l'ébauche d'un tel mécanisme en combinant le placage de déplacements avec un *bump mapping*. Ce mécanisme permet de déterminer si le placage de déplacements doit être réellement effectué et à quel degré de précision, ou si seulement un *bump mapping* peut être appliqué. L'ensemble de ces calculs s'effectuant directement sur le processeur graphique, la transition entre les différents niveaux de détails peut ainsi être gérer par un traitement au niveau des pixels, ce qui permet d'éviter les changements brusques lors du passage d'un niveau de détails à l'autre. Cette approche permet de moduler les performances de rendu des différentes méthodes en fonction de différents paramètres. Cette approche nous semble intéressante et est étendue dans le cadre des travaux décrits dans cette thèse.

Chapitre 4

Visualisation de reliefs de faibles amplitudes

Dans ce chapitre nous décrivons la méthode que nous avons développée pour visualiser en temps réel et de façon réaliste des reliefs n'étant composés que de faibles amplitudes, comparés à la taille de la surface d'un objet. Comme nous avons choisi de nous intéresser à des peintures numérisées pour donner un cadre applicatif à ces travaux, la méthode présentée, dans ce chapitre, est donc dédiée à la visualisation d'objets dont la base peut être associée à un plan. L'application de cette méthode de visualisation à des tableaux numérisés, est décrite en détails au chapitre 5.

Au cours de ce chapitre, nous présenterons donc en premier lieu, à la section 1 l'approche retenue pour visualiser ces données, puis nous décrivons, en détails, cette méthode au cours de la section 2, avant de présenter des premiers résultats à la section 3. Ce chapitre est finalement ponctué par un bilan de cette méthode de visualisation à la section 4.

1 Présentation de l'approche hybride

Comme mentionné au chapitre 2, notre technique de visualisation doit traiter des reliefs dont l'amplitude est assez faible, en regard de la taille de l'objet visualisé, par exemple de l'ordre de quelques millimètres pour un objet représentant une surface de l'ordre du mètre carré. Le relief doit ainsi être modéré par rapport à l'échelle de la macrostructure. En considérant cette échelle de reliefs, en fonction des conditions de visualisation, l'ajout réel de ces derniers n'est pas toujours nécessaire. Ils peuvent parfois être simplement simulés. C'est l'une des raisons qui nous a orientés vers le développement d'une approche hy-

bride de rendu, combinant une simple simulation des reliefs et l'ajout réel de ceux-ci. Il est donc nécessaire, dans un premier temps, de revenir sur ce type d'approches, avant de décrire plus en détail celle que nous avons développée.

1.1 Principe des approches hybrides pour l'ajout de détails

Dans le cadre des méthodes d'ajout de détails, un certain nombre de travaux sont basés sur des combinaisons de différentes représentations associées à des niveaux d'échelles des détails. Par exemple, Tatarchuk [Tat06a] propose simplement de combiner *relief mapping* et *bump mapping*, en remplaçant l'ajout réel du relief par sa simulation, quand celui-ci devient peu visible (*i.e.* occupe moins d'espace à l'écran). Seul un critère de distance est utilisé pour commander le passage d'une représentation à une autre.

Nous avons également vu, au cours du chapitre d'état de l'art précédent, qu'une méthode plus aboutie a été proposée par Becker et Max [BM93]. Pour cette technique, les auteurs proposent de combiner trois représentations différentes des détails, en combinant trois techniques de rendu différentes : fonctions de réflectance bi-directionnelle, *bump mapping* et placage de déplacements. Dans ces travaux, Becker et Max posent le cadre nécessaire pour effectuer des transitions continues entre les différentes techniques, notamment en redistribuant les vecteurs normaux pour garder une cohérence entre les trois techniques de rendu et en définissant un placage de déplacements partiel pour garder des transitions douces et progressives, notamment entre le *bump mapping* et le placage de déplacements. Le choix du passage d'une méthode de rendu à une autre est fonction d'un coefficient dépendant, entre autres de la distance à la caméra et de l'angle formé par le vecteur de vue et le vecteur normal de la surface de l'objet (*i.e.* représente la direction d'observation). Du fait de l'utilisation d'un critère de sélection principalement basé sur la distance au point de vue et non pas sur les données, l'objectif final de cette méthode peut ainsi se résumer à proposer une solution aux problèmes liés à l'aliasing des surfaces rendues via un algorithme de *bump mapping* et s'éloigne, de ce fait, des objectifs recherchés par notre technique de rendu.

Ces travaux sont, en partie, à la base des méthodes qui ont pour objectif d'offrir une modulation du niveau de précision avec lequel les détails de la surface d'un objet sont rendus. On peut, par exemple, citer les travaux de Olano *et al.* [OKS03] qui proposent une technique de niveaux de détails automatiques, permettant de moduler la complexité d'un programme de shaders appliqué à une surface (en particulier pour rendre une *BRDF*, suivant plusieurs niveaux de précision). D'autres méthodes sont également dédiées à traiter des cas bien spécifiques comme, par exemple, les vagues d'océans avec les travaux

de Bruneton *et al.* [BNH10].

Néanmoins, le principal problème des méthodes hybrides résident dans les transitions entre les différents modes de rendu, qui sont souvent enclines à générer des artefacts visuels. Comme le souligne Heidrich *et al.* [HDKS00], les artefacts les plus importants proviennent des différences d'illumination entre les différents algorithmes. Généralement, une composition des différents rendus avec des niveaux de transparences progressifs (parfois améliorée, comme dans les travaux de Giel et Wimmer [GW07]) permet d'éviter les artefacts liés aux changements brusques de géométrie ou d'illumination (*popping artefacts*) en proposant des transitions progressives. Ces problèmes mis à part, les méthodes hybrides restent des outils puissants pour accélérer le processus de rendu. Cependant, il n'existe pas de solutions générales : chaque technique est adaptée à une situation spécifique. C'est pourquoi nous avons choisi d'introduire notre propre méthode, spécialement conçue pour le rendu des reliefs de faibles amplitudes, comme, par exemple, ceux formés par la peinture des toiles numérisées, le crépi d'un mur ou encore d'un sol parsemé de cailloux et de gravillons.

1.2 Principe général de la visualisation hybride de reliefs à faibles amplitudes

La méthode proposée dans ce chapitre est basée sur le constat suivant : la mésostructure des objets que nous souhaitons visualiser, présentent souvent des caractéristiques géométriques fines qui méritent d'être préservées lors de leur rendu. Une partie des reliefs qui forment cette surface, sont, en effet, généralement significatifs d'un point de vue visuel. Mais la grande majorité de ces reliefs ont une amplitude qui reste très souvent assez faible, par rapport à la macrostructure, et est, par conséquent, négligeable d'un point de vue géométrique. Ces différents reliefs peuvent donc être approximés par une technique de rendu simplifiée, afin d'augmenter la vitesse de rendu de ces objets. C'est pourquoi nous proposons de découpler le relief, dont la géométrie est significativement perceptible, du reste des données.

L'objectif de cette méthode est ainsi de rendre le plus rapidement possible ces objets planaires, tout en minimisant au maximum les défauts visuels. Pour cela, nous proposons d'effectuer le rendu de leur mésostructure, en combinant dynamiquement une technique de rendu fidèle des zones de reliefs les plus significatifs, d'un point de vue géométrique, avec une méthode de rendu simplifiée qui approxime les zones de la mésostructure où les reliefs sont négligeables [1].

1. Il faut garder à l'esprit que les notions de reliefs significatifs et négligeables, utilisées dans la description de cette méthode, sont toujours exprimées d'un point de vue géométrique,

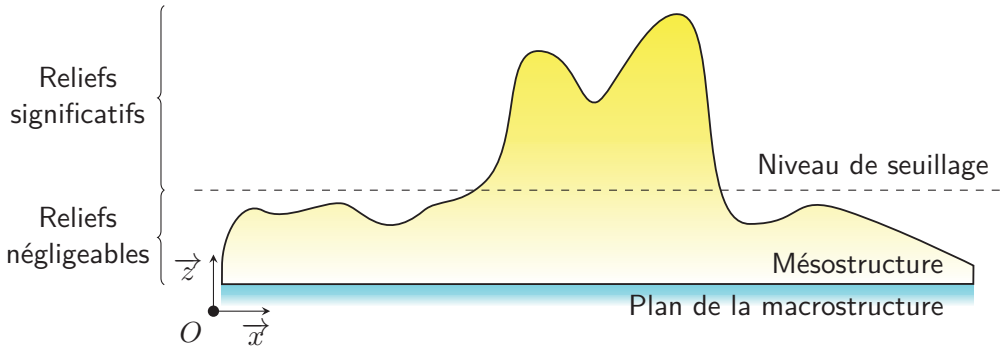


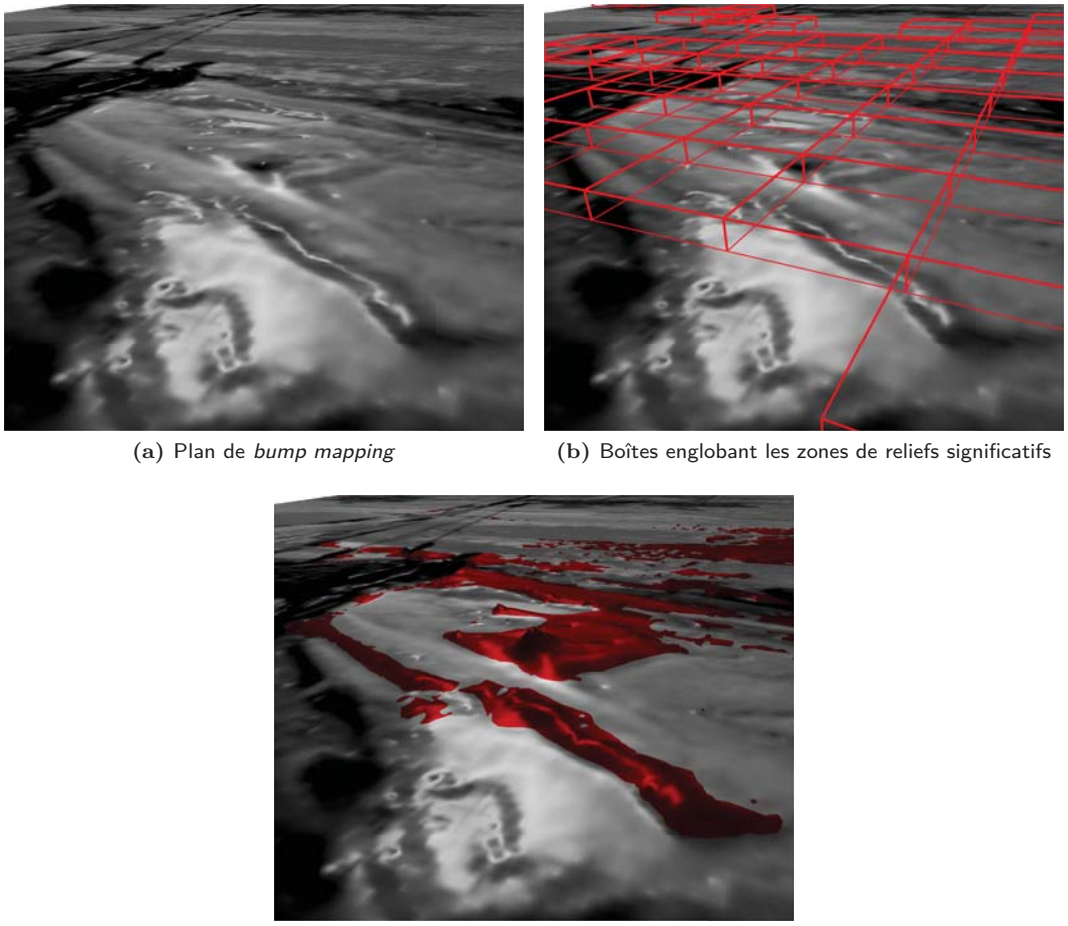
Figure 4.1 – Classification du relief : deux catégories de reliefs en fonction d'une valeur de seuil appliquée sur l'altitude des données.

Toutefois, pour obtenir des résultats visuels satisfaisants, la combinaison de deux méthodes de rendu ne peut se limiter à une sélection par un mécanisme simple, uniquement basé sur un critère de distance à la caméra. Il faut, en effet, prendre en compte les données formant les reliefs de la mésostructure pour en optimiser le rendu. C'est en partie grâce à ce point précis que nous nous démarquons des méthodes présentées à la section précédente.

Le relief formant la mésostructure, est alors classifié par un seuillage des élévations contenues dans le champ de hauteurs, comme l'illustre la figure 4.1. Les points situés en dessous du seuil forment les reliefs négligeables, d'un point de vue géométrique. Ils sont affichés à l'aide d'un simple algorithme de *bump mapping*. Par opposition à cette première classe de reliefs, ceux assimilés aux reliefs significatifs, toujours d'un point de vue géométrique, sont représentés par un ensemble de boîtes les englobants. Situées au dessus du plan de la macrostructure, elles sont affichées à l'aide d'un rendu de reliefs plus complexe, ayant recours à un algorithme de lancer de rayons. Afin d'obtenir des performances temps réel de rendu, cet algorithme est exécuté par le processeur graphique. La fusion des deux techniques est gérée par un mécanisme adaptatif, basée sur une heuristique qui permet d'accélérer de manière significative la vitesse de rendu.

Un rendu couleur de qualité de l'objet est finalement obtenu grâce à une description des variations spatiales de son matériau. Ces variations, qui correspondent aux variations de la réflectance à la surface de la mésostructure, sont représentées par une texture de réflectance bi-directionnelle qui encode, en chaque texel, un lobe spéculaire definit par le modèle de Lafortune [LFTG97], tel que défini à la section 3 page 20. Cette information est généralement estimée à partir d'un ensemble de mesures de réflectances effectuées durant une phase

par rapport à l'échelle de la macrostructure. La taille de ces reliefs reste donc dans des valeurs faibles d'élévations, même pour les reliefs dits significatifs.



(a) Plan de *bump mapping*

(b) Boîtes englobant les zones de reliefs significatifs

(c) Reliefs significatifs réellement ajoutés (en rouge)

Figure 4.2 – Illustration de l'application de la méthode proposée : le plan de base est rendu utilisant un *bump mapping* (a), sur lequel est rendu un ensemble de boîtes englobant (b) les zones de reliefs significatifs (c).

de numérisation (*cf.* chapitre 5). Le rendu couleur est évalué, à la volée, par le matériel graphique, pendant la phase de rendu.

Au final, pour visualiser ces objets, nous utilisons simplement un champ de hauteurs pour représenter la géométrie de la mésostructure. Pour représenter la couleur, un ensemble de textures couleurs est utilisé, dans lequel est encodée l'information de réflectance, suivant le modèle de Lafortune.

2 Méthode de rendu hybride et adaptative

Comme décrit précédemment, le relief du champ de hauteurs est classifié dans deux catégories distinctes, en fonction de l'impact de sa géométrie sur le rendu final de la mésostructure. Ces deux classes de reliefs déterminent deux sortes de reliefs, comme l'illustre la figure 4.1, les reliefs *négligeables*, dont la géométrie n'est que visuellement peu perceptible sur le rendu, et les reliefs *significatifs*, dont la géométrie est visuellement plus importante. Chacune de ces classes de reliefs est affichée à l'aide d'une technique de rendu différente : le *bump mapping* pour les reliefs négligeables (*cf.* figure 4.2a) et une technique dérivée des algorithmes de placage de reliefs pour les reliefs significatifs (*cf.* 4.2c). En effet, il est connu que le *bump mapping* entraîne des erreurs de parallaxe lorsque les reliefs qu'il simule sont trop importants et sont visuellement perceptibles. Dans la méthode proposée, nous éliminons ces erreurs en remplaçant la simulation du relief par un ajout réel de celui-ci.

Ces deux techniques sont fusionnées et combinées grâce à un mécanisme qui choisit automatiquement le meilleur algorithme, selon les conditions courantes d'observation et les caractéristiques des données, accélérant ainsi le rendu. La prise en compte des données, pour sélectionner localement la méthode de rendu à employer, nous démarque ainsi des méthodes précédentes et notamment de celle que propose Tatarchuk [Tat06b] qui n'utilisent généralement qu'un critère de distance et parfois d'angle de vue.

Le relief significatif est représenté par un ensemble de boîtes, généré au cours d'une phase de pré-traitements des données. Ces boîtes sont construites dans le but de définir un volume englobant les zones où le relief est significatif, comme illustré par la figure 4.2b. Les données qui représentent le relief étant sous forme de champ de hauteurs, elles sont stockées dans une texture. Dans cette carte de hauteurs, sont également inclus les vecteurs normaux définis en chaque point de la carte d'élévations. Le calcul des normales est effectué en considérant l'ensemble des triangles qu'il est possible de construire avec le 8-voisinage d'un point dans la carte d'élévations. Pour garder une certaine compacité de ces données, seules les composantes représentant l'abscisse et l'ordonnée de ces vecteurs sont stockées, la troisième coordonnée étant recalculée à la volée. Cette texture est utilisée pour le rendu des deux classes de reliefs.

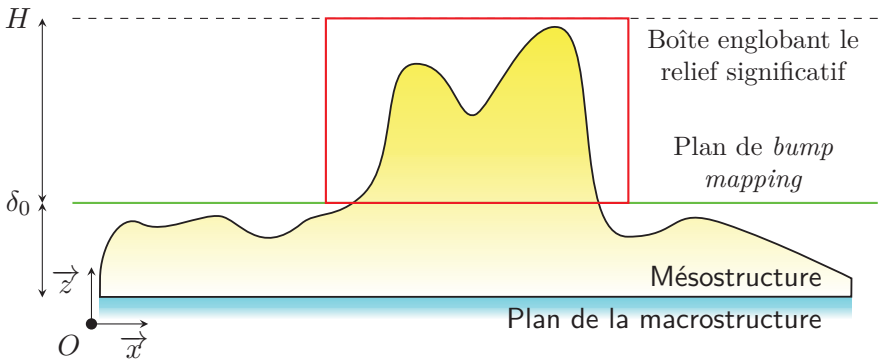


Figure 4.3 – Position relative du plan représentant la macrostructure (en vert) et des boîtes englobant le relief significatif. La boîte matérialisée en rouge contient une partie de ce relief significatif tandis que le relief négligeable se trouve sous le plan de *bump mapping*.

2.1 Rendu du relief négligeable

Étant donné la nature planaire des objets visualisés, la surface de la macrostructure est approximée par un simple plan. Sur ce plan, les reliefs négligeables sont rendus en utilisant un algorithme de *bump mapping* classique. Comme la partie significative du relief ne commence qu'au delà d'un niveau de seuillage δ_0, un espace existe entre la hauteur minimale de la macrostructure et la hauteur de la partie basse des boîtes englobant les zones de relief significatif. Pour palier à ce problème et ainsi éviter d'éventuel trous dans la surface rendue de la mésostructure, le plan de *bump mapping* est translaté verticalement jusqu'à ce seuil. Par conséquent, et comme l'illustre la figure 4.3, la vraie toile se trouve en fait sous le plan de *bump mapping*. Cependant, la hauteur maximale de la mésostructure reste la même et l'amplitude de son relief reste ainsi totalement respecté.

2.2 Rendu du relief significatif

L'algorithme de rendu utilisé pour les zones de relief significatifs est une variante des algorithmes de placage de détails décrits au chapitre 3. Plus précisément, l'algorithme utilisé est basé sur celui du *relief mapping* décrit par Policarpo *et al.* [POC05]. Afin de maintenir la continuité avec les reliefs négligeables, les zones rendues à l'aide de cet algorithme sont superposées directement sur le plan servant à appliquer l'algorithme de *bump mapping*. Cela permet en même temps de corriger les erreurs de parallaxe introduites par ce dernier lorsque le relief devient trop important.

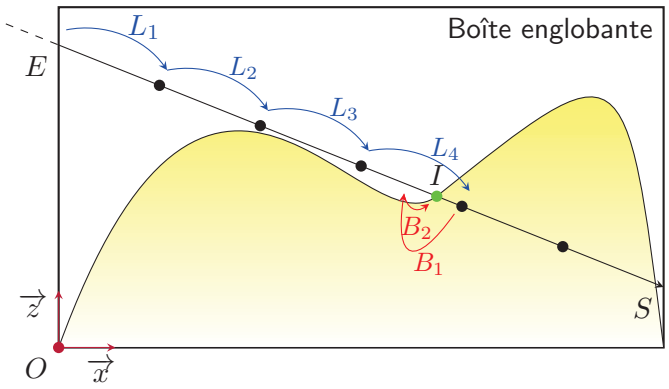


Figure 4.4 – Méthode de recherche d'intersections utilisée entre un rayon et le champ de hauteurs représentant la mésostructure. Une recherche linéaire (pas en bleu) est combinée avec une recherche dichotomique (pas en rouge).

L'algorithme de lancer de rayons utilisé pour ces travaux, est très proche de celui du *relief mapping* et n'en diffère que de quelques points. Comme l'illustre la figure 4.4, l'algorithme de recherche combine une étape de recherche linéaire avec une phase de recherche dichotomique destinée à affiner la précision de l'intersection retournée. Ces étapes sont représentées respectivement par les pas L_i et B_j de la figure 4.4. Toute cette phase de recherche d'intersections est entièrement exécutée par le processeur graphique.

Habituellement, l'algorithme du *relief mapping* utilise un simple polygone comme support de rendu, la surface du relief étant ajoutée sous la surface de ce polygone. À l'opposé de l'approche standard, notre méthode utilise des boîtes définies dans un espace à trois dimensions. Chacune de ces boîtes est définie par sa position (u_b, v_b) et sa taille (w_b, h_b), exprimée dans l'espace texture correspondant au champ de hauteurs. Ainsi, chaque boîte est associée à une portion de la texture, qui est, par conséquent, partagée entre toutes les boîtes. Les hauteurs minimales et maximales sont identiques pour toutes les boîtes, et correspondent respectivement au niveau du plan de *bump mapping* et à la hauteur H qui est définie par l'élévation maximale de la carte de hauteurs représentant la mésostructure, *cf.* figure 4.3.

Pour un rendu efficace de l'algorithme de lancer de rayons à partir de boîtes, nous associons des coordonnées de texture à trois dimensions à chaque sommet afin de définir un repère local propre à chaque boîte (Ox et Oz sur la figure 4.4) et à l'intérieur duquel le lancer de rayons est effectué. Au cours du rendu, ces coordonnées de texture fournissent immédiatement le point d'entrée

du rayon dans la boîte, quelqu'en soit la face considérée. L'algorithme de lancer de rayons procède alors ainsi : pour chaque pixel atteint par la boîte actuellement rendue, les points d'entrée E et de sortie S du rayon d'observation sont déterminés à l'aide des coordonnées de texture à trois dimensions. Ces deux points donnent le rayon de traversée sur lequel la recherche d'intersection avec le champ de hauteurs doit être faite. Les phases de recherches linéaire et binaire sont alors effectuées de la même manière que pour le *relief mapping* standard, c'est-à-dire en combinant une recherche linéaire définie pour un nombre fixe de pas et une recherche dichotomique, définie elle aussi avec un nombre fixe de pas.

2.3 Mécanisme de rendu adaptatif

Étant donné l'amplitude des reliefs des mésostructures considérées, lorsqu'on l'observe selon une direction proche de la normale au plan de base de la macrostructure (*i.e.* avec un angle de vue perpendiculaire au plan de base de la macrostructure), elle ne présente quasiment aucune différence visuelle entre les reliefs significatifs et les reliefs négligeables. Cette constatation est également valable lorsque le point de vue est éloigné de la surface de l'objet, en d'autres termes, lorsque les variations de hauteurs deviennent négligeables par rapport à leur taille projetée à l'écran. En effet, dans ces conditions de visualisation, la distance entre la surface réelle et le plan de *bump mapping* devient alors trop petite pour être visible.

Pour tirer parti de ces différents constats, nous avons développé une heuristique qui permet de déterminer automatiquement si une zone de relief significatif doit être réellement visualisée ou simplement simulée, c'est-à-dire si la boîte l'englobant doit être affichée ou non. Comme toutes méthodes hybrides, la notre peut potentiellement être sujette aux artéfacts de *popping* qui surviennent lors d'un passage brusque d'une méthode de rendu à l'autre. Dans le but d'éviter ces problèmes, une composition de transparence est effectuée. Pour cela, la transition, entre la simulation et l'ajout réel du relief, est faite en dessinant le relief mais en lui appliquant un effet de transparence. Cela permet de faire des transitions progressives entre les deux modes de rendu, les parties de relief significatif apparaissant ou disparaissant progressivement (*cf.* figure 4.8 page 69). Le rôle du mécanisme adaptatif est donc de déterminer pour chaque boîte englobant le relief significatif son état, c'est-à-dire si celle-ci est visible ou non, et si oui, son niveau de transparence.

L'état d'une boîte est donné par un coefficient α_{box} calculé à partir du point I, de plus haute altitude du relief au sein de la boîte. Ses coordonnées sont exprimées en espace texture correspondant au champ de hauteurs qui

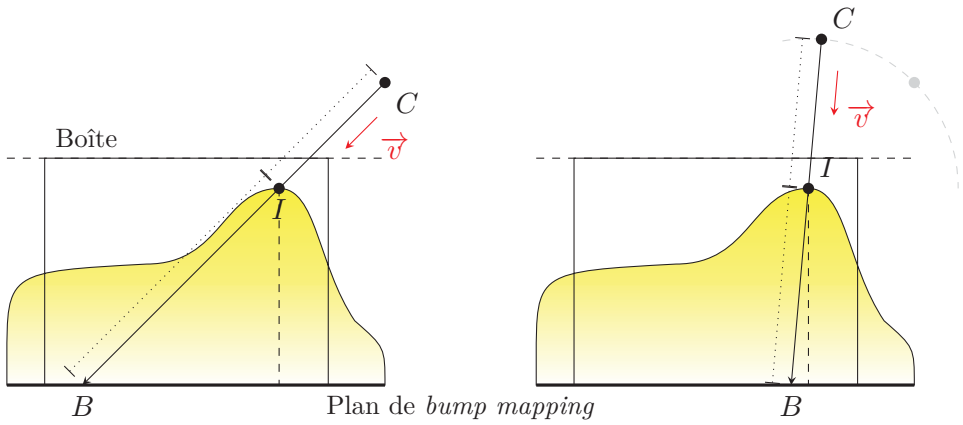


Figure 4.5 – Calcul du coefficient de transition α_{box} à partir du point I le plus haut de chaque boîte et de la position C de la caméra. Le ratio $\|\overrightarrow{IB}\|/\|\overrightarrow{CI}\|$ dépend de l'angle de vue et de la distance à la surface.

représente la géométrie de la mésostructure. Elles sont déterminées lors de la phase de construction des différentes boîtes. Durant cette phase, sont également calculées la valeur h_{max}, définissant l'élévation maximale du relief de l'ensemble de la boîte, et les coordonnées (u_{max}, v_{max}) qui permettent de repérer la position de ce point sur la macrostructure. Dans le cas de l'application de cette méthode de rendu aux tableaux numérisés, les différentes étapes qui permettent d'extraire ces informations, sont décrites lors de la présentation de notre protocole de numérisation, au chapitre 5.

Ainsi, en considérant la projection B du point I sur le plan de *bump mapping* le long de la direction d'observation $\vec{v}$, nous pouvons voir que le ratio r définit par :

$$r = \frac{\|\overrightarrow{IB}\|}{\|\overrightarrow{CI}\|}$$

où C représente la position de la caméra, dépend directement de l'angle de vue, mais également de la distance au point de vue, comme l'illustre la figure 4.5. Ce ratio tend vers zéro pour des points de vue éloignés ou frontaux, et augmente lorsque l'on se rapproche ou que l'angle de vue devient rasant. On peut en déduire que ce ratio est pertinent et peut servir de base à notre mécanisme adaptatif. Par conséquent, le calcul du coefficient α_{box} est défini en s'appuyant sur ce ratio et est donné par l'équation 4.1 :

$$\alpha_{box} = \begin{cases} 0 & \text{si } K \cdot r < \varepsilon_{min} \\ 1 & \text{si } K \cdot r > \varepsilon_{max} \\ \frac{K \cdot r - \varepsilon_{min}}{\varepsilon_{max} - \varepsilon_{min}} & \text{sinon} \end{cases} \tag{4.1}$$

où ε_{min} et ε_{max} sont deux valeurs de seuil et K est un facteur, défini par

l'utilisateur, permettant de contrôler la sensibilité de la transition entre les deux algorithmes de rendu. Le calcul du coefficient α_{box} est effectué séparément pour chacune des zones de relief significatif, délimitées par les boîtes, et est évalué pour chaque image rendue.

Selon la valeur de α_{box}, il y a finalement trois cas à considérer lors du rendu de chaque boîte englobante :

- Si α_{box} est nul, la boîte n'est pas dessinée. Correspond au cas où le relief significatif peut n'être que simulé.
- Si $\alpha_{box} = 1$, la boîte est affichée sans composition α. Correspond au cas où le relief significatif doit être visualisé réellement.
- Si $\alpha_{box} \in]0, 1[$, la boîte est composée avec le plan de *bump mapping*. Correspond à la transition entre les deux états précédents, permettant d'éviter les changements brusques de rendu.

De part ce mécanisme et ce calcul du coefficient α_{box}, les transitions sont implicitement gérées par le troisième cas considéré pour les valeurs du paramètre α_{box}.

2.4 Illumination

L'objectif de cette méthode étant de fournir un mécanisme de rendu réaliste, nous y avons donc inclus un algorithme d'illumination offrant une modélisation réaliste de l'illumination. Elle s'appuie notamment sur une méthode capable d'exploiter des mesures bi-directionnelles de la réflectance de la surface de la mésostructure. On peut toutefois noter qu'un modèle plus simple, tel que le modèle de Phong, peut évidemment être utilisé.

Une fois, le type de rendu sélectionné par le mécanisme adaptatif pour une zone de la mésostructure donnée et, le cas échéant, une fois le relief ajouté, le pixel courant peut être coloré en évaluant son ombrage. Cette évaluation est effectuée en utilisant un modèle de Lafortune [LFTG97] pour calculer l'illumination. Pour l'estimer, nous utilisons l'approximation proposée par McAllister *et al.* [MLH02] à un seule lobe spéculaire, qui peut être définie par l'équation 4.2 déduite à partir de l'équation générale 2.1 page 21 du modèle de Lafortune :

$$f_r(\vec{v}, \vec{l}) = \rho_d \ + \ \rho_s \ (C_x v_x l_x + C_y v_y l_y + C_z v_z l_z)^k \tag{4.2}$$

Dans cette expression les variables ρ_d et ρ_s sont des vecteurs représentant une couleur à trois composantes, décrivant respectivement les contributions diffuses et spéculaires à l'énergie lumineuse sortante. Les vecteurs $\vec{v}$ et $\vec{l}$ correspondent respectivement aux directions locales d'observation et d'incidence lumineuse.

Ce modèle a été choisi pour sa simplicité, ce qui lui permet d'être évalué

directement et efficacement par le matériel graphique, tout en offrant des résultats visuels satisfaisant. Il a également été retenu pour sa compacité (avec un seul lobe spéculaire notamment), étant donné que seules trois textures à deux dimensions sont nécessaires pour le représenter en mémoire : deux textures sont utilisées pour stocker ρ_d et ρ_s et la troisième permet d'encoder les paramètres (C_x, C_y, C_z, k) qui décrivent la forme du lobe spéculaire.

L'équation 4.2 est donc directement évaluée par la carte graphique à partir de l'information de couleur qui est disponible via deux textures RGB et une RGBA, stockées dans la mémoire graphique. Les directions d'illumination et d'observation sont toutes les deux corrigées à l'aide de l'élévation présente dans le champ de hauteurs, et cela même pour le *bump mapping*. De cette manière, on peut prendre en compte la vraie position de la surface dans le calcul d'ombrage. Les vecteurs normaux intervenant dans cette étape sont recalculés à la volée à partir des coordonnées n_x et n_y estimées précédemment et stockées avec le champ de hauteurs.

3 Résultats

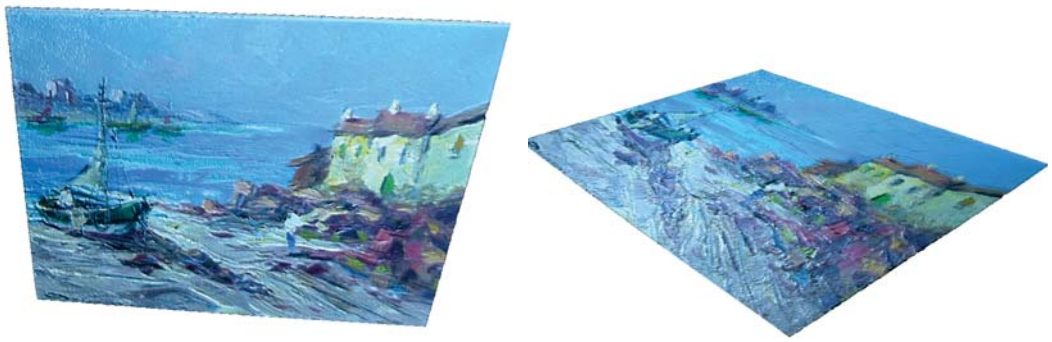
Afin de donner un aperçu des performances de notre méthode de visualisation, nous donnons quelques résultats de tests effectués sur la méthode proposée, notamment dans le cas de la visualisation des reliefs formés par la peinture d'une toile numérisée.

3.1 Qualité visuelle

Comme nous l'avons décrit à la section précédente, le calcul d'illumination final est confié à un modèle de Lafortune, limité à un seul lobe spéculaire. Ce modèle permet de représenter la réflectance de la mésostructure de façon réaliste, tout en gardant une consommation mémoire raisonnable. Afin de démontrer l'efficacité de cette méthode, nous l'avons donc appliquée à la visualisation de tableaux numérisés (la méthode de numérisation employée est décrite au chapitre 5). Ainsi, la figure 4.6 propose un comparatif visuel des résultats obtenus par notre méthode de rendu par rapport à la toile réelle.

Généralement, dans le cas des méthodes hybrides combinant plusieurs algorithmes évaluant l'illumination de la surface, des inconsistances lumineuses peuvent apparaître. Dans notre cas, la même normale, la même information de couleur et le même modèle d'illumination sont utilisés pour les deux algorithmes de rendu. Ainsi, aucune différence d'illumination n'est visible au niveau

(a) Tableau réel

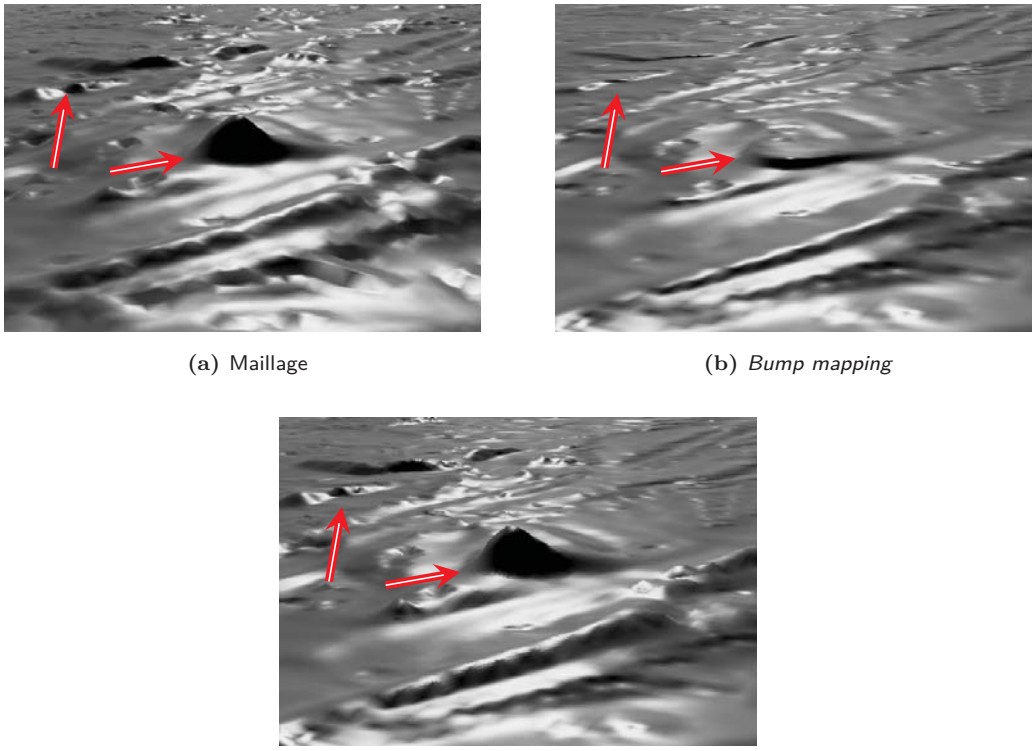


(b) Rendu obtenu

Figure 4.6 – Comparaison entre le tableau réel (a) et le rendu obtenu par notre méthode (b) pour des positions de point de vue et des conditions d'éclairage assez proches.

des transitions entre les différentes zones de relief et lors du passage d'un algorithme de rendu à l'autre. Un exemple est donné par la figure 4.7 qui compare les rendu obtenus pour différentes techniques de rendu : un rendu de référence avec un maillage construit à la résolution des données, un simple *bump mapping* et la méthode proposée dans ce chapitre. On constate également que les erreurs de parallaxe les plus fortes, introduites par le *bump mapping*, sont correctement corrigées pour les reliefs les plus significatifs, grâce à l'utilisation du placage de déplacements par pixels.

Le mécanisme, permettant de sélectionner la technique de rendu à employer pour les différentes parties des données qui ont un relief signgificatif, a également été développé pour limiter au maximum les apparitions soudaines de géométrie, en faisant apparaître et disparaître progressivement la géométrie réellement ajoutée. Par conséquent, cette catégorie d'artéfacts de rendu est totalement absente de notre méthode.



(a) Maillage **(b)** *Bump mapping*

(c) Rendu hybride

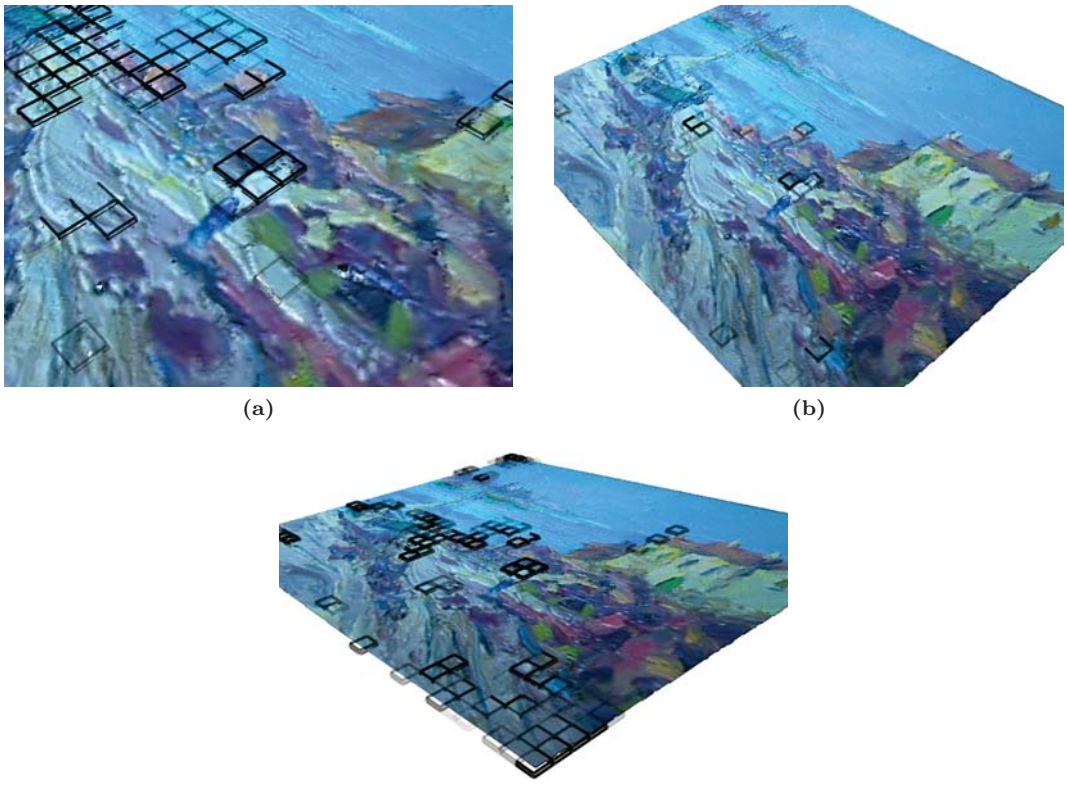
Figure 4.7 – Comparaison entre un rendu de référence par un maillage (a), via un simple *bump mapping* (b) et notre rendu hybride (c) sur un tableau numérisé. Les erreurs de parallaxe introduites par le *bump mapping* sont corrigées par l'utilisation de placage de déplacements, là où le relief devient significatif. Le calcul d'illumination reste donc constant entre ces différents rendus.

3.2 Performances de rendu

Les performances obtenues avec notre méthode hybride sont reportées dans la tableau 4.1, dans la colonne *Hybride + Adaptatif*. Dans ce tableau, les performances de rendu sont comparées avec différentes techniques de rendu : rendu par un maillage à pleine résolution des données (rendu de référence), *relief mapping* appliqué sur l'ensemble de la carte d'élévations représentant le tableau, et enfin notre approche hybride sans le mécanisme adaptatif (*i.e.* les boîtes du relief significatif sont toujours rendues) et avec le mécanisme adaptatif. On peut tout de suite constater que la méthode proposée permet d'atteindre une visualisation temps réel du tableau numérisé. Les fréquences d'affichage ont été mesurées à l'aide d'un ordinateur PC standard, équipé d'un processeur *AMD Athlon X2 4200+* et doté d'une carte graphique *NVIDIA*

Distance	Angle de vue	Maillage	*Relief mapping*	Hybride	Hybride + Adaptatif
proche	rasant	71	36	33-48	36-57
	face	85	32	39-48	56-72
cadré	rasant	60	100	61	330
	face	52	74	60	350
loin	rasant	96	280-450	82	650
	face	96	155-230	80	1250

Tableau 4.1 – Vitesse d'affichage (en Hz) pour le Tableau du Port en utilisant différents algorithmes de rendu et pour différentes conditions d'observation.



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

Figure 4.8 – Comportement du mécanisme adaptatif : pour les gros plans (a) et les angles de vue rasant (c) , un nombre plus important de boîtes englobant le relief significatif est automatiquement sélectionné afin de compenser les erreurs de parallaxe introduites par le *bump mapping*.

Taille des boîtes	Nombre de boîtes	Angle de vue		
		Face	Intermédiaire	Rasant
5×5	2073	305	142	18
10×10	714	300	185	38
20×20	253	330	250	75
50×50	81	240	175	145
100×100	35	135	130	125

Tableau 4.2 – Vitesse de rendu (en Hz) de notre méthode hybride adaptative pour différentes tailles de boîte (exprimées en pixels) et différents angles d'observation.

GeForce 7900GTX.

Sur ce tableau 4.1, on peut voir que l'accélération apportée par l'adaptativité est significative. En effet cette technique surclasse systématiquement le *relief mapping* standard appliqué sur l'ensemble du champ de hauteurs. Il faut toutefois nuancer ces résultats par le fait que notre approche est moins efficace qu'un rendu de maillages, mais uniquement pour des points de vue très proches. En effet, la vitesse du rendu adaptatif dépend fortement du point de vue. À des angles rasants ou en gros plans, le relief est beaucoup plus visible et l'adaptativité requière plus de boîtes, comme le montre la figure 4.8, ce qui diminue d'autant les performances. Ce mécanisme dépend aussi du nombre total de boîtes, qui est directement lié à leur taille. Pour une petite taille, les boîtes approximent bien le relief mais le calcul de visibilité par boîte devient plus coûteux. Au contraire, une taille de boîte plus grande entraîne un nombre de pixels plus important à rendre par le biais de l'algorithme de lancer de rayons qui est bien plus coûteux que le simple *bump mapping*. Afin d'évaluer l'impact de la taille des boîtes, les résultats de tests de performance sont donnés dans le tableau 4.2, pour différentes tailles de boîte.

Un des avantages à utiliser un algorithme proche du *relief mapping* est la correction des erreurs de parallaxe introduites par la simulation du relief via un *bump mapping*, comme le montre la figure 4.7. Néanmoins, il est important de remarquer que le rendu de champs de hauteurs texturés impose certaines limitations. En effet, si un saut important se produit entre les hauteurs de deux pixels adjacents, la couleur le long de la falaise générée est simplement interpolée entre ces deux points. Ainsi, plus les discontinuités dans le relief sont importantes, plus la texture est étirée. Cette limitation rend difficile l'extension de notre méthode au rendu de reliefs plus importants, bien que McGuire et Whitson [MW08] proposent une solution à ce type de problèmes. Un autre problème du *relief mapping* est, comme le soulignent entre autres Baboud et Décoret [BD06], la présence d'artefacts dus à l'échantillonnage discret du

rayon lors de la recherche d'intersections. Dans le cas des objets ici, le relief est vraiment faible par rapport à ce qui est généralement rendu à l'aide de cet algorithme, et par conséquent ces artefacts se trouvent assez limités. Cela nous permet par ailleurs de réduire le nombre de pas nécessaire à la recherche linéaire afin d'accélérer encore un peu plus la vitesse rendu.

4 Conclusion

Au cours de ce chapitre, nous avons présenté la technique que nous avons développée afin de visualiser des reliefs de faibles amplitudes sur des objets planaires. Bien que les mésostructures ciblées par cette méthode ne soient composées que de reliefs de faibles amplitudes par rapport à la taille des objet, cette technique de visualisation propose de prendre en compte, au moment du rendu, les reliefs les plus significatifs d'un point vue géométrique et visuel. À ces fins, cette méthode développe une approche hybride combinant un rendu par la technique du *bump mapping*, associé à un placage de déplacements par pixels. Ce dernier est effectué sous la forme d'un algorithme de lancer de rayons, exécuté par le processeur graphique, et est dérivé du *relief mapping*. Les deux algorithmes de rendu sont sélectionnés alternativement via un mécanisme adaptatif. Il s'appuie sur une heuristique développée pour prendre en compte différents paramètres de visualisation, dont la position de l'observateur mais également la nature du relief qui compose la mésostructure.

L'application finale que nous avons faite de cette méthode a été la visualisation de tableaux numérisés. Cependant, afin d'obtenir les données représentant ces œuvres d'art, il est nécessaire de développer un certain nombre d'outils permettant de créer ces copies virtuelles de tableaux. C'est pourquoi, nous avons développé une chaîne complète de traitements permettant de les numériser en vue de leur visualisation, via la méthode hybride décrite dans ce chapitre. Cette chaîne de traitements est présentée en détails dans le chapitre suivant. Elle a ainsi permis de fournir des données réelles nécessaires à l'évaluation de la méthode de rendu développée. Cette évaluation détaillée est présentée au chapitre 5.

Chapitre 5

Chaîne de traitements pour la numérisation de tableaux

Dans ce chapitre, nous décrivons le protocole mis en place pour numériser des tableaux. Ce protocole a été élaboré en parallèle du développement de la méthode de rendu présentée au chapitre précédent, afin de garder une chaîne de traitements relativement simple et efficace, en adéquation avec la méthode de visualisation.

Tout au long de ce chapitre, après une brève présentation et un bref rappel des objectifs de ce protocole de numérisation à la section 1, les différentes étapes qui composent cette chaîne de traitements, sont décrites en détails, à la section 2. Différents résultats expérimentaux menés sur la chaîne de traitements sont présentés à la section 3, avant de dresser un premier bilan de ces travaux, au cours de la section 4.

1 Présentation du protocole de numérisation d'un tableau

L'objectif initial de ce protocole est de fournir des données qui soient exploitables par la technique de visualisation de reliefs de faibles amplitudes présentée précédemment. L'originalité de cette méthode de visualisation étant la prise en compte, au moment du rendu, du relief de la mésostructure, elle nécessite un protocole de numérisation adéquate afin de produire l'information de relief nécessaire. L'autre point important est l'acquisition de l'information de réflectance, variant sur l'ensemble de la surface de la toile qui doit pouvoir être adaptée au modèle d'illumination retenu, à savoir le modèle de Lafortune et plus précisément son approximation via un seul lobe spéculaire [MLH02]. Par conséquent, cette chaîne de traitements doit inclure tous les pré-traitements

qui sont nécessaires aux données pour les adapter, de manière optimale, à leur visualisation temps réel et réaliste.

Le méthode proposée est assez flexible et est adaptée à la numérisation de tous types d'objets planaires dont la surface est formée par des reliefs de faibles amplitudes, par rapport à la taille de l'objet. Ce protocole se rapproche ainsi de certaines techniques d'acquisition récemment développées pour numériser des textures de réflectance [RK09]. Toutefois, comme nous l'avons mentionné dans le chapitre 2, nous nous sommes focalisés sur la numérisation de tableaux d'art. En effet, cette classe d'objets se prêtent parfaitement à la représentation retenue, du fait de la forme planaire de la toile, sur laquelle ne se trouvent que les reliefs formés par la peinture, qui restent faibles au regard de la taille du tableau.

Une contrainte supplémentaire que nous nous sommes fixés, concerne le déroulement même du protocole. En effet, nous avons souhaité que ce protocole de numérisation, ainsi que les traitements qu'il met en jeu, restent simples afin qu'ils puissent être effectués par des personnes non-initiées à ces thématiques précises de numérisation et de traitements de ces données.

Pour résumer les objectifs techniques de ce protocole, les données que doit fournir cette chaîne de traitements à la méthode de rendu sont les suivantes :

- les données géométriques représentant le relief de la peinture sur la toile, sous forme d'un champ de hauteurs incluant les vecteurs normaux de la surface de la peinture ;
- la classification du relief en deux catégories distinctes : zones de relief négligeable et zones de relief significatif ;
- un ensemble de boîtes englobant les zones de relief significatif et pour chacune des boîtes, le point d'altitude maximale ainsi que sa position sur la toile ;
- les textures représentant la réflectance de la toile encodée via une texture de réflectance bi-directionnelle.

Pour générer cet ensemble de données, nous proposons que la chaîne de traitements soit découpée en différentes étapes successives :

1. acquisition, en trois dimensions, de la géométrie du tableau ;
2. extraction de la géométrie représentant la toile de son environnement de numérisation (cadre, ...) et génération du champ de hauteurs ;
3. classification du relief et création de l'ensemble des boîtes englobant les zones où le relief est significatif ;
4. acquisition de la réflectance ;
5. encodage de la texture de réflectance bi-directionnelle.

Ces différentes étapes nécessitent des dispositifs matériels dédiés pour effectuer les acquisitions de la géométrie et de la réflectance. Ceux-ci sont présentés et décrits en détail dans les sections suivantes.

2 Description de la chaîne de traitements

La première étape de ce protocole de numérisation (section 2.1), est consacré à la numérisation de la géométrie, et plus particulièrement celle formée par le relief de la peinture. Une fois ces données acquises, elles sont traitées afin de générer le champ de hauteurs et d'effectuer la classification des différentes zones de reliefs, à partir de laquelle est extrait l'ensemble des boîtes servant au rendu du relief significatif (section 2.2). La seconde partie du protocole (section 2.3), quant à elle, se concentre sur l'acquisition et le traitement des données permettant de représenter la couleur de la toile, et en particulier, sa réflectance.

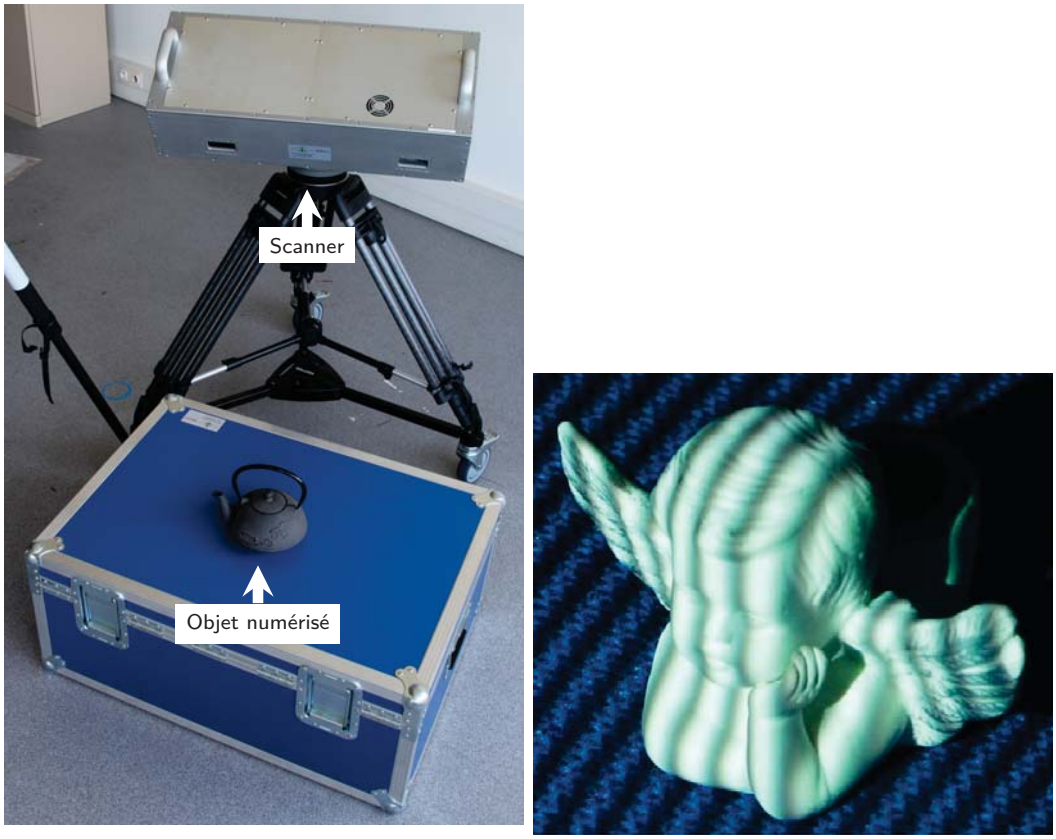
2.1 Acquisition de la géométrie

Pour acquérir la géométrie de la peinture, nous avons utilisé un scanner à trois dimensions, fonctionnant sur le principe de la lumière structurée. Ce dispositif matériel ayant l'avantage de fournir des données sous forme d'*images télémétriques* (*range images*), il est assez simple d'en extraire une représentation du relief de la toile sous forme de champ de hauteurs.

Le protocole défini dans cette section fait suite aux travaux développés dans la thèse de Frédéric Larue [Lar08]. En effet, ces travaux sur l'acquisition d'objets réels ont servi de base au protocole proposé pour numériser les tableaux.

Description du matériel utilisé Le matériel utilisé, illustré par la figure 5.1a se limite à un scanner à lumière structurée. Ce scanner fonctionne sur le principe suivant : un vidéo-projecteur projette, sur l'objet à numériser (*cf.* figure 5.1b), plusieurs séries de franges lumineuses (une alternance de bandes de lumière blanche et de bandes sans lumière) dont la taille et la position varie suivant un décalage pré-défini et connu du scanner.

En se projetant sur l'objet, ces différentes franges sont déformées et les déformations engendrées sont enregistrées par une caméra située à une distance connue précisément du vidéo-projecteur. Cela permet de mesurer les déformations induites sur les franges via un calcul basé sur une triangulation optique.



(a) Dispositif de numérisation par un scanner à lumière structurée

(b) Franges lumineuses projetées par le scanner dont les déformations sur l'objet permettent d'en définir sa géométrie

Figure 5.1 – Numérisation par lumière structurée.

Le décalage de chaque frange étant connu et effectué précisément, il est ainsi possible d'associer à chaque frange une paramétrisation à une dimension de la surface, suivant une orientation orthogonale aux franges. Une paramétrisation à deux dimensions peut facilement être étendue à partir de la surface couverte par l'ensemble des franges projetées sur l'objet. De cette manière, le scanner peut fournir une image télémétrique, où chaque pixel représente la distance entre la surface de l'objet et le dispositif de numérisation.

Protocole de numérisation de la géométrie La mesure de la géométrie, effectuée par notre scanner à trois dimensions, s'appuie sur une mesure du contraste entre les franges lumineuses qu'il projette. Par conséquent, étant donné que les détails chromatiques de la peinture peuvent présenter de forts contrastes, notre protocole de numérisation est basé sur une capture de plusieurs cartes 3D, chacune effectuée avec différents temps d'exposition. Cela nous permet de garantir la bonne acquisition des régions claires et sombres.

Du fait de la forme très particulière de la classe d'objets que nous nous proposons de numériser, l'acquisition de l'information 3D peut être limitée à un seul point de vue. Celui-ci est choisi perpendiculairement au plan de la toile. De cette façon, la paramétrisation 2D de la carte 3D et l'angle de vue quasi-orthogonal utilisé nous fournissent une représentation géométrique de la peinture sous une forme déjà très proche du champ de hauteurs que nous désirons obtenir, comme l'illustre la géométrie brute obtenue après numérisation d'un objet réel, sur la figure 5.2a. Il reste cependant à extraire ce dernier des données obtenues.

2.2 Traitement des données géométrique spécifiques aux toiles

Afin d'exploiter les données numérisées, en vue de leur visualisation, il convient de les traiter pour pouvoir les exploiter de façon optimale. Plusieurs étapes de traitements sont nécessaires. Il faut, dans un premier temps, extraire la toile de son environnement pour générer la carte d'élévations représentant la surface de la peinture. À partir de cette carte de hauteurs, le relief doit être classifié selon les deux catégories prises en compte au moment du rendu. Et enfin, il faut créer l'ensemble des boîtes englobant le relief le plus significatif pour qu'il puisse être rendu via l'algorithme de lancer de rayons, décrit au chapitre précédent.

a) Extraction automatique de la toile

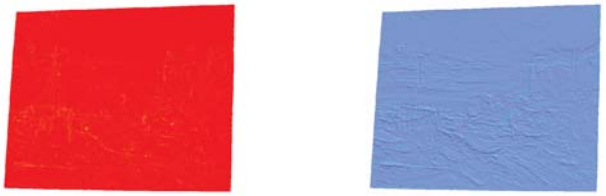
La numérisation de la géométrie de la toile s'effectue avec son environnement immédiat qui, par exemple, peut inclure son cadre, comme c'est le cas sur la figure 5.2a. Par conséquent, des données géométriques représentant cet environnement sont présentes sur les images télémétriques. Selon l'application, on peut vouloir supprimer l'information qui ne concerne pas directement la surface peinte. Il est alors intéressant de pouvoir disposer d'un traitement permettant d'extraire uniquement la toile en ignorant par exemple le cadre et toute autre information géométrique non pertinente.

La particularité d'un tableau numérisé est la forme hautement planaire de sa géométrie. En effet, son relief peut être vu comme une variation de hauteurs par rapport à un plan de référence. Par conséquent, pour récupérer le champ de hauteurs correspondant au tableau, nous effectuons une analyse en composantes principales de la carte 3D afin d'y extraire un nouveau repère local $(\vec{X}, \vec{Y}, \vec{Z})$, où $\vec{X}$ et $\vec{Y}$ sont les axes de dispersion principale, tangents au plan de la toile, et où $\vec{Z}$ est l'axe orthogonal. Les positions des points de la carte 3D sont alors recalculées par rapport à ce nouveau repère, et la coordonnée

(a) Géométrie brute après numérisation.

(b) Calcul d'un premier champ de hauteurs par analyse en composantes principales

(c) Application d'un filtre de Sobel pour repérer la transition entre le cadre et la toile.



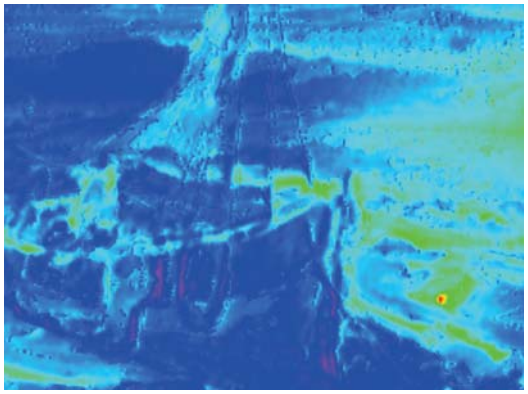
(d) Extraction de la toile par un algorithme de croissance de région.

(e) Champ de hauteurs (avec les vecteurs normaux) résultant d'une nouvelle application d'une analyse en composantes principales.

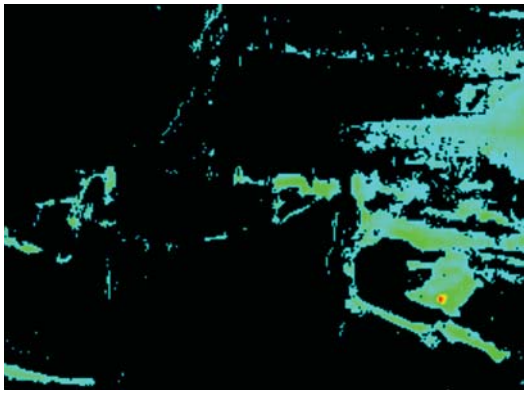
Figure 5.2 – Étapes de traitements permettant d'extraire le champ de hauteurs représentant la toile.

associée à l'axe $\vec{Z}$ permet ainsi de représenter la coordonnée de hauteur (*cf.* figure 5.2b).

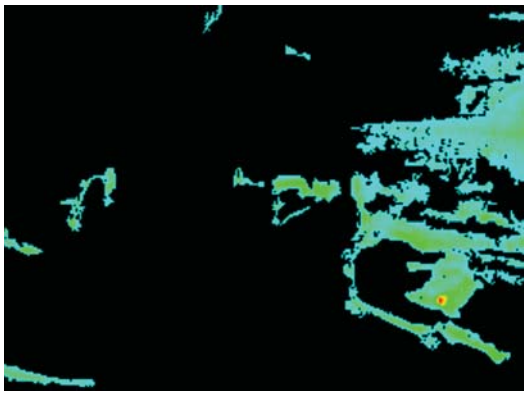
En considérant les données géométriques dans ce nouveau repère, la transition entre la toile et le cadre provoque nécessairement une discontinuité importante de la hauteur comparée aux petits détails géométriques qui forment la surface de la toile et de la peinture. Ainsi, nous appliquons l'opérateur de Sobel sur les coordonnées de hauteurs afin de mettre en valeur ces fortes discontinuités (*cf.* figure 5.2c). Pour détecter la partie intérieure de la peinture qui correspond à la toile, nous appliquons un algorithme de croissance de région qui s'arrête lorsqu'une zone de gradient trop important est rencontrée. Nous choisissons comme point de départ de la croissance le centre de la carte d'élévations, puisqu'il nous semble évident qu'un tableau correctement mesuré se trouve bien centré dans le champ du dispositif d'acquisition. Seule la région résultant de ce remplissage est conservée, comme l'illustre la figure 5.2d, ignorant ainsi les points qui n'appartiennent pas à la toile. Une érosion de quelques pixels est ensuite appliquée afin d'éviter la présence d'artefacts résiduels. Une fois que seule l'information pertinente demeure, une nouvelle analyse en com-



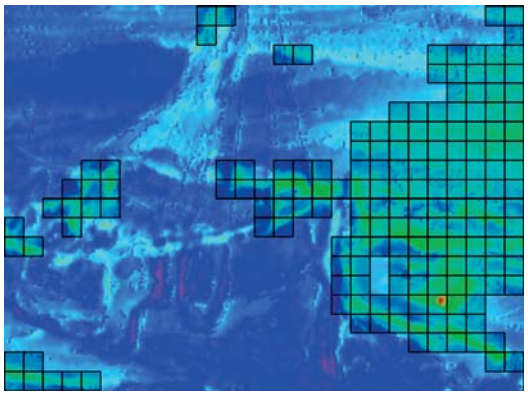
(a) Carte de hauteurs résultant de l'alignement par analyse en composantes principales de l'image télémétrique après extraction de la toile.



(b) Seuillage des coordonnées de hauteurs. Seules restent les zones de relief significatif.



(c) Nettoyage permettant de réduire l'influence du bruit.



(d) Division cellulaire des composantes connexes restantes, permettant de définir l'ensemble de boîtes utilisé pour le rendu du relief significatif.

Figure 5.3 – Étapes de classification du relief entre reliefs négligeables et reliefs significatifs. Le relief est mis en évidence par un gradient de couleurs.

posantes principales est appliquée pour recalculer un repère local $(\vec{X}, \vec{Y}, \vec{Z})$ qui soit plus représentatif des seules données de la toile. Le champ de hauteurs représentant la surface de la peinture peut ainsi être construit à partir des coordonnées de hauteurs des points ré-exprimées dans ce nouveau repère. On obtient ainsi des données semblables à la figure 5.2e.

b) Classification du relief de la peinture

Notre but est maintenant d'extraire la partie du relief de la toile (*cf.* figure 5.3a) qui est suffisamment significative, d'un point de vue visuel. Cette opération est menée en effectuant un seuillage sur les élévations du champ de hauteurs. Pour cela, nous définissons un seuil δ_0 en dessous duquel la hauteur

est considérée comme négligeable. À l'heure actuelle, ce seuil est fourni manuellement. Durant nos tests, nous avons fixé sa valeur entre 20% et 40% de la hauteur maximale de la toile. Pour effectuer la classification, tous les pixels de la carte d'élévations dont la hauteur est supérieure au seuil δ_0 sont marqués comme étant *valides*, et les autres comme étant *non valides*. De cette façon, on construit un masque binaire $\mathcal{M}_v$, dont les résultats sont visibles sur la figure 5.3b. Pour limiter les trop petites zones de reliefs marquées comme valides, les composantes connexes de $\mathcal{M}_v$ qui sont trop réduites (*i.e.* dont la taille ne dépasse pas quelques pixels) sont supprimées (*cf.* figure 5.3c). Cela permet, entre autres, d'ignorer la possible influence du bruit de numérisation.

La grille de pixels de la carte de hauteurs est ensuite subdivisée en un ensemble de cellules de $n \times n$ pixels. Cette taille de cellule permet de définir la taille des boîtes utilisées pour le rendu du relief, et par conséquent, la valeur de n représente un paramètre important pour notre algorithme de rendu hybride. En effet, cette valeur permet de contrôler la granularité du mécanisme adaptatif. Seules les cellules contenant les reliefs les plus élevés doivent être conservées, par conséquent, les cellules qui ne contiennent aucun pixel valide dans $\mathcal{M}_v$ sont supprimées. Pour les autres, une boîte est créée, dont les dimensions de sa base sont définies par la taille n en pixel de la cellule. Leur hauteur est fixée par l'altitude maximale des reliefs de la toile. La figure 5.3d donne un exemple de la division cellulaire obtenue sur notre tableau de test.

Une fois cette étape effectuée, le relief segmenté est alors totalement inclus dans l'ensemble final de boîtes, comme l'illustre la figure 5.3d. Cet ensemble de boîtes constitue l'ensemble des boîtes utilisé pour rendre séparément la partie significative du relief. Comme notre mécanisme adaptatif nécessite de connaître le point le plus haut de chaque cellule, nous calculons également, pour chaque boîte, la hauteur h_{max} et les coordonnées, exprimées dans l'espace du champ de hauteurs, (u_{max}, v_{max}) de son point le plus élevé. Il est évident, mais important, de noter que plus n est petit, plus il y aura de boîtes qui seront créées. Le choix de cette valeur est discuté plus en détails dans la section présentant les résultats expérimentaux (*cf.* section 3).

Après ces différentes étapes, toute l'information géométrique nécessaire au rendu est compactée de façon à être exploitée par le processeur graphique lors de la visualisation. Pour cela, les vecteurs normaux de la surface sont calculés à partir du champ de hauteurs puis sont stockés, avec la carte d'élévations, dans une texture à quatre composantes. Les données définissant les boîtes (coordonnées, position du point le plus élevé) sont compactées dans plusieurs tableaux de sommets.

2.3 Acquisition de la réflectance

La seconde partie du protocole de numérisation est dédiée à l'acquisition de la "couleur" de la toile. Celle-ci est faite par l'acquisition des effets de réflectance bi-directionnelle qui sont par la suite encodée au sein d'une approximation du modèle de Lafortune, comme décrit au chapitre 4. En vue de générer ce modèle, il faut obtenir un échantillonnage assez dense de la réflectance de la toile, pour différentes positions d'éclairage et d'observation. Pour cela, l'information chromatique de la toile est capturée à l'aide d'un ensemble de photographies, prises pour des points de vue et des positions d'éclairages différents. Toute la difficulté de ce processus consiste à repérer les positions relatives de la source lumineuse et de la caméra, par rapport au tableau numérisé.

a) Protocole d'acquisition des échantillons de réflectance

Afin de repérer précisément la position de la source lumineuse, nous utilisons une structure mécanique rigide permettant de couvrir l'ensemble de l'hémisphère supérieur de l'objet. Cette structure est présentée par la figure 5.4. Elle se compose d'un chariot mobile supportant la source lumineuse[1]. Celui-ci se déplace suivant un arceau décrivant un demi-cercle sur lequel plusieurs positions clés ont été repérées (*cf.* figure 5.4c). L'objet à numériser est, pour sa part, placé sur une table rotative. Ainsi, en combinant les déplacements de la source lumineuse et de l'objet, il est possible de couvrir toutes les directions d'éclairage situées sur un hémisphère.

Pour localiser précisément la source lumineuse, aux différentes positions clés, par rapport au tableau, lors d'une phase de calibrage, celle-ci est remplacée sur le chariot mobile par une caméra. Nous utilisons la méthode décrite par Larue et Dischler [LD06] utilisant une paramétrisation par lumière structurée pour repérer la position de la caméra, et, par conséquent, déterminer la position de la source lumineuse pour l'ensemble des positions clés. Il faut toutefois noter l'impossibilité de garantir que la position de la caméra coïncide exactement avec celle de la source lumineuse. Cependant, la structure rigide ayant été développée dans ce sens, on peut supposer que l'erreur commise ne sera que de quelques millimètres ce qui, reporté à l'échelle des distances séparant l'objet de la caméra (environ 1 mètre), peut être considérée comme négligeable.

1. Pour ces travaux, nous avons utilisé une lampe à incandescence *Jocker-Bug 200* de la marque *K5600 lighting*. Ce type de lampe présente un filament très fin permettant de s'approcher d'une source lumineuse ponctuelle et de générer ainsi des ombres très marquées.

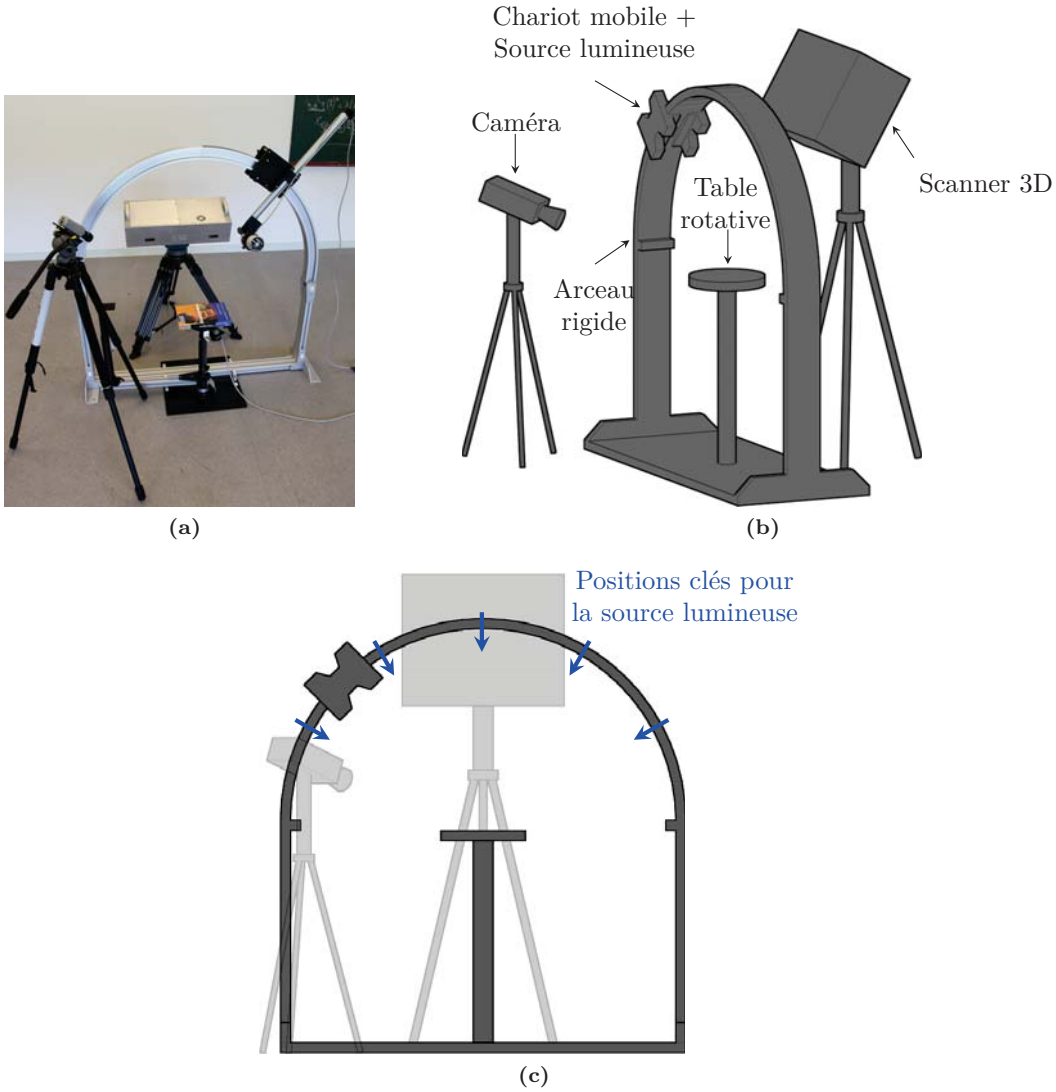


Figure 5.4 – Banc de numérisation utilisé pour acquérir la réflectance.

Une fois cette étape de calibration effectuée, l'acquisition à proprement parler de la réflectance peut avoir lieu. Pour cela, la caméra est placée à différentes positions qui combinées avec la rotation de la table supportant le tableau permet de couvrir un large échantillon de directions d'observation. Chaque position de la caméra, par rapport à la toile, est déterminée par la méthode utilisée précédemment [LD06]. Une image couleur de la toile est ensuite prise pour chaque combinaison de direction d'éclairage et d'angle de vue. Comme pour chaque image, chacune de ces combinaisons est connue, les photographies faites peuvent ainsi être recalées pour correspondre à la géométrie de la peinture. De façon similaire, les positions de la source lumineuse peuvent également être exprimées par rapport à cette même géométrie.

b) Extraction du modèle de Lafortune

Une fois les photographies recalées sur la géométrie, des échantillons de luminance sont extraits pour chaque point de la carte télémétrique par projection dans l'espace image de chaque photographie. Ces échantillons sont composés des directions locales d'observation et d'illumination ainsi que l'information de luminance du pixel atteint. Parmi ces multiples échantillons, certains sont sujets aux auto-occlusions. Celles-ci se produisent lorsque la lumière est stoppée par la géométrie avant d'arriver en un point ou lorsque certains points ne sont pas visibles pour un point de vue donné. Certains modèles analytiques de représentation de textures bi-directionnelles sont sensibles, en termes de précision, aux fortes discontinuités que peuvent représenter de tels effets d'ombre. Cependant, comme dans notre cas, l'information de relief est connue, nous choisissons donc d'éliminer les échantillons qui ne correspondent pas à de l'illumination ou à l'observation directe. Ces différents échantillons sont détectés par un algorithme de lancer de rayons. On peut noter que cette phase est facultative et ne peut avoir lieu que lorsque la peinture toile présente de forts empâtements.

Un modèle de Lafortune est finalement estimé en chaque point de la carte de hauteurs. Comme mentionné dans le chapitre précédent (*cf.* équation 4.2 page 65), nous utilisons l'approximation de Mc Allister *et al.* [MLH02] à un lobe spéculaire pour générer ce modèle. On peut toutefois noter que ce modèle de réflectance, bien que bi-directionnelle, ne s'apparente pas à une *BRDF*, dans le sens où les mesures de luminance effectuées prennent également en compte les inter-réflexions de la peinture. Il ne s'agit pas non plus d'une mesure de *BTF*, car dans notre cas, la géométrie de mésostructure est explicitement connue et utilisée pour le rendu.

3 Résultats expérimentaux

Nous avons testé ce protocole de numérisation sur le Tableau du Port. Cette toile est illustrée sur la figure 5.5 et également par la figure 4.6 page 67, qui propose une comparaison entre des photographies de l'original et des rendus de notre copie numérique avec un angle de vue et un éclairage assez proches. L'évaluation de la méthode de visualisation étant faite au chapitre précédent, nous ne nous limiterons ici qu'à quelques commentaires qui permettent de mettre en lumière certaines caractéristiques et performances de notre chaîne de traitements.

Figure 5.5 – Tableau utilisé pour tester le protocole de numérisation.

3.1 Acquisition

Le test de la chaîne de traitements a été mené sur un tableau de petites dimensions (environ $25,5cm \times 21cm$). Le champ de hauteurs finale extrait de cette numérisation, et après extraction de la toile, représente une taille de 731×594 échantillons d'élévations. La numérisation complète du tableau s'est déroulée pendant une journée entière.

Pour l'acquisition de la réflectance, afin de fournir un échantillonnage suffisamment dense de points de vue et de positions d'éclairage, nous avons acquis 339 images pour différents combinaisons d'angles de vue et de positions de la source lumineuse. Étant donné la configuration de notre structure rigide, nous avons utilisé 8 orientations différentes du plateau supportant le tableau. À chaque orientation, un ensemble de mesures ont été faites pour différentes positions de caméra (entre 6 et 8 positions différentes de caméra, *cf.* figure 5.6) et en utilisant, pour chaque point de vue, 5 positions d'éclairage différentes (la figure 5.7 donne un exemple de ces différentes position pour une orientation et une position de caméra données.).

Figure 5.6 – Différentes positions de caméra utilisées pour une orientation donnée de la toile.

3.2 Occupation mémoire

Au niveau de l'occupation mémoire des données générées pour les copies virtuelles, le principal avantage de notre méthode est l'absence de maillage pour représenter la géométrie, ce qui engendre un faible coût mémoire. En fait, la carte d'élévations capturée par le scanner est directement traitée et utilisée comme champ de hauteurs. Les différents traitements qui composent notre chaîne travaillent donc directement sur l'échantillonnage géométrique initial.

Pour stocker les données qui correspondent au tableau de la figure 5.5, seules quatre textures flottantes à 16 bits (une RGBA et trois RGB) ainsi qu'une liste de boîtes, sont nécessaires pour représenter la copie numérique finale, incluant la géométrie de la peinture et la texture de réflectance bi-directionnelle. Ces textures ont une taille de 731×594 pixels. La taille totale après traitement est d'environ 11 Mo, dont moins de 900 ko sont dédiés à la

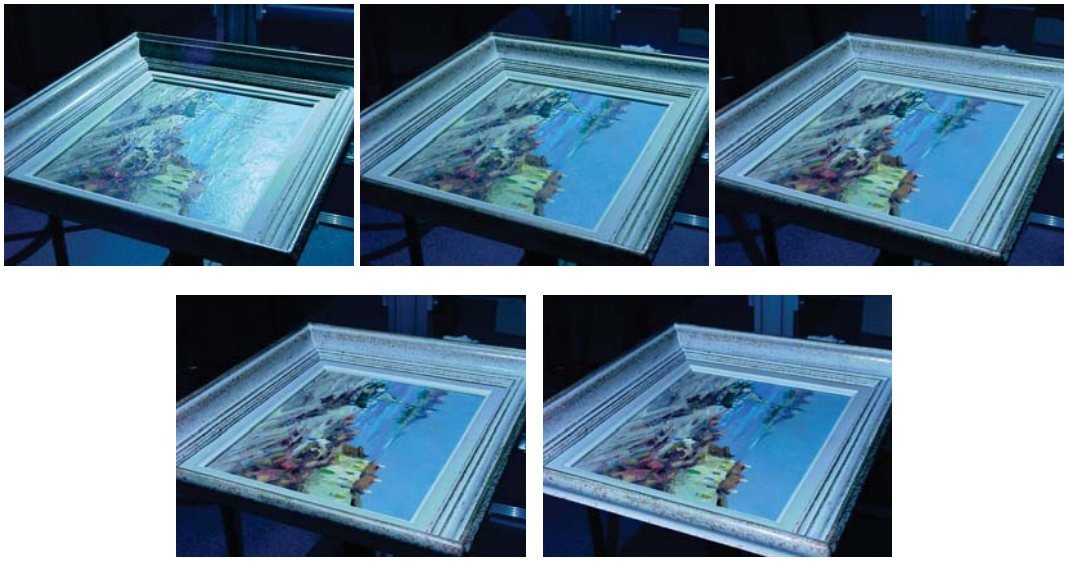


Figure 5.7 – Positions de la source lumineuse pour une orientation de la toile et un position de la caméra données.

géométrie. À titre de comparaison, on peut noter que le maillage correspondant à la géométrie de ce tableau contient environ 815k triangles pour environ 434k sommets ce qui représente environ 5 Mo, uniquement pour stocker la position des sommets.

Comme les données finales sont simplement sous forme de textures couleurs, on peut imaginer que ces données puissent être compressées via des algorithmes de compression standard destinés aux images tels que la compression *JPEG*, par exemple. On pourrait ainsi réduire encore la taille de ces données, afin d'autoriser le transfert de ces données à travers Internet, pour, par exemple, construire des galeries de peintures virtuelles.

4 Conclusion

Au cours de ce chapitre, et plus généralement de cette partie, nous avons présentés une technique permettant d'acquérir puis de visualiser des tableaux d'art. Au travers de ces objets spécifiques, nous avons présenté une chaîne de traitements permettant à la fois d'acquérir des objets pour en extraire une représentation sous forme de données à deux dimensions et demie et de les préparer à une visualisation réaliste en temps réel. L'acquisition et le rendu des données prend également en compte l'information colorimétrique des objets numérisés, via une représentation de la réflectance bi-directionnelle sous

la forme d'un modèle analytique. Grâce à ces travaux, nous avons apporté une méthode nouvelle pour gérer toute la phase d'acquisition des données géométriques et colorimétriques, ce qui inclut également tous les traitements appliqués aux données numérisées. Au niveau de la visualisation, nous avons proposé une méthode de visualisation de tableaux qui ne s'appuie pas uniquement sur l'information de couleur mais qui prend également en compte les reliefs de la surface à rendre afin d'en améliorer les résultats visuels. Dans le but d'obtenir une méthode de rendu performante, nous avons choisi de suivre une approche hybride, combinant un algorithme de lancer de rayons GPU et une simulation du relief via un algorithme de *bump mapping*, attelée à un mécanisme adaptatif de sélection du type de rendu. La combinaison de ces deux modes de rendu permet de limiter les artéfacts de rendu, notamment celles introduites par le *bump mapping*, tout en conservant des performances temps réel de visualisation. Il est important de noter que ce mécanisme adaptatif utilise un critère de choix qui est basé sur les conditions de visualisation, et également sur la nature des reliefs visualisés, en tenant compte de leur amplitude.

La validation de cette chaîne de traitements a été effectuée sur une toile dont la taille est assez restreinte, ce qui lui permet d'entrer entièrement dans les champs d'acquisition de notre scanner à lumière structurée et de notre structure rigide d'acquisition de la réflectance. Aussi, il paraît assez évident que ces travaux pourrait être étendus afin de gérer l'acquisition de tableaux d'envergure plus importante, n'entrant pas entièrement dans les champs de vision du scanner et de la structure rigide, ce qui pose actuellement des problèmes au niveau de certains traitements, et notamment pour l'étape de numérisation de la réflectance.

Un point qui mérite une certaine attention, est cependant apparu sur les résultats expérimentaux. En effet, le rendu via un maillage se montre plus rapide dans le cas où le champ de hauteurs est observé de très près. Par conséquent, il pourrait être avantageux d'intégrer ce mode de rendu au mécanisme adaptatif, pour les points de vue très proches de la surface de la toile. De même, dans le cas où la toile est regardée avec un point de vue très éloigné, il pourrait être judicieux d'étudier l'intégration d'un mécanisme de filtrage sur les données, de type pyramide de *mipmaps*, qui permettrait dans ce cas de figure de limiter d'éventuels effets d'aliasing.

Il peut être également intéressant de noter que les données géométriques utilisées dans le cadre de cette méthode sont basées sur une structuration très simple : un simple champ de hauteurs. Par conséquent, aucune structure de données complexe à calculer n'est employée. Les données géométriques sont même directement exploitées par les algorithmes de rendu sans contraintes quant à une représentation statique de celle-ci : rien n'interdit de modifier

les données durant le rendu. Évidemment, dans ce cas de figure, une texture de réflectance bi-directionnelle ne pourrait plus être exploitée du fait des précalculs nécessaires à sa création.

Ces travaux ont également été menés, avec pour objectif, de montrer la viabilité d'une technique hybride dans le cadre de la visualisation de données à deux dimensions et demie. Par ce cas pratique, sur des données où l'échelle du relief est faible (de l'ordre de quelques millimètres), on a pu montrer qu'il est possible d'obtenir des performances de rendu élevées, aussi bien en terme de qualité visuelle que de vitesse, tout en développant une technique simple de visualisation. Il est également apparu que le maillage reste une technique efficace dans certains cas d'utilisation bien précis, et cela, même si les algorithmes de lancer de rayons sont particulièrement bien adaptés à ce type de données, et peuvent être exécutés directement sur le processeur graphique. Par conséquent, il est assez naturel que nous ayons tenté de transcrire ce type d'approches hybrides à la visualisation de données à deux dimensions et demie, mais toujours en essayant de garder une représentation simple des données, comme cela peut être le cas avec les copies virtuelles de tableaux. Cette étude fait l'objet de la seconde partie de cette thèse et est dédiée à la visualisation de données dont la représentation reste des champs de hauteurs mais dont les reliefs de leur surface possède des amplitudes qui peuvent être beaucoup plus élevées (de l'ordre de plusieurs centaines de mètres voire du kilomètre). Cette approche est particulièrement dédiée à la visualisation de données topographiques représentant des terrains.

Deuxième partie

Visualisation de terrains

Chapitre 6

Visualisation de terrains

Dans la seconde partie de cette thèse, nous proposons une méthode de visualisation de données à deux dimensions et demie, dont les données peuvent former des reliefs d'amplitudes assez fortes, c'est-à-dire à une échelle plus importante que dans le cas des tableaux. Cette méthode est particulièrement adaptée à la visualisation de données topographiques. Au cours de ce chapitre, nous décrivons plus spécialement les principales méthodes développées jusqu'à présent, qui sont dédiées à la visualisation de terrains.

1 Introduction

Les données qui ont fait l'objet de la première partie de cette thèse, ont, pour caractéristique principale, une faible amplitude des reliefs qu'elles représentent, par rapport à la taille de la macrostructure. Les données traitées dans cette seconde partie présentent des reliefs qui deviennent plus importants. Ainsi, la géométrie qui est représentée par la surface du champ de hauteurs, est visuellement toujours importante, à l'échelle de distances avec laquelle les données sont généralement observées. Par conséquent, une simple simulation de ceux-ci, telle qu'effectuée dans le cas des reliefs de la surface de la peinture de tableaux, n'est plus envisageable. Cela constitue un problème couramment rencontré en informatique graphique.

Ainsi, depuis quelques décennies, la visualisation de données à deux dimensions et demie est activement étudiée dans la communauté de l'informatique graphique. Elle prend très souvent la forme de visualisation de terrains. Ainsi, un grand nombre de méthodes de visualisation de terrains ont été développées. La majorité de ces méthodes utilisent des données topographiques représentées, à l'origine, sous forme de champs de hauteurs.

Historiquement, les méthodes de visualisation de terrains utilisent une représentation différente des données pour en effectuer la visualisation. Ces méthodes transforment la représentation initiale des données, le plus souvent, en maillages. Ces maillages sont extraits à partir du champ de hauteurs et se composent généralement d'un très grand nombres de polygones (de plusieurs millions à plusieurs centaines de millions, suivant les jeux de données). Par conséquent, il est nécessaire d'adapter ces maillages aux conditions de visualisation afin d'en optimiser la vitesse et la qualité de rendu. À ces fins, plusieurs approches pour la construction et la gestion des maillages ont été proposées, mettant en œuvre des mécanismes de niveaux de détails (*levels-of-details, LOD*) construits par des étapes de simplifications successives des données. Une des approches possibles consiste en l'application de méthodes génériques développées pour tous types d'objets représentés par un maillage [DFKP05] et, en particulier, pour les objets composés d'un très grand nombre de triangles (comparable à un terrain). Appliquées à la visualisation d'objets à deux dimensions et demie, ce type de méthodes ne se montre pas suffisamment efficace. En effet, cette structuration des données très particulière est caractérisée par une grande régularité des données qui peut être mise à profit pour en optimiser la visualisation. Par conséquent, ces méthodes générales ne sont pas détaillées dans cet état de l'art. Nous nous concentrons uniquement, dans ce chapitre, sur les méthodes de visualisation, plus performantes, spécialement dédiées aux terrains qui sont décrites dans les sections 2, pour les méthodes orientées CPU et 3, pour les méthodes orientées GPU.

Depuis l'avènement des processeurs graphiques massivement parallèles et programmables, de nouvelles approches se développent pour le rendu interactif de terrains. Elles sont basées sur des algorithmes de lancer de rayons exécutés directement sur le processeur graphique. L'utilisation du matériel graphique permet d'envisager leur emploi, non plus uniquement pour du rendu non interactif, mais également pour du rendu interactif, voire temps réel. Cette catégorie de méthodes fait l'objet d'une étude détaillée dans la section 4.

Comme mentionné précédemment, les champs de hauteurs contiennent généralement une très grande quantité d'échantillons. La gestion (mémoire en particulier) de cette masse de données est critique. Cependant, nous ne traitons dans cette section que des techniques de visualisation, les mécanismes de gestion de la masse de données (mécanismes d'accès disques, de streaming sur des réseaux, etc.) ne sont pas abordés.

2 Maillages multi-résolutions

La représentation la plus courante des terrains est le maillage de triangles. Cependant, le nombre de triangles nécessaire pour représenter un terrain est très important. Par conséquent, il est nécessaire de mettre en place des mécanismes qui permettent de limiter le nombre de triangles à rastériser. C'est dans cette optique que différentes représentations multi-résolutions ont été développées. Certaines de ces approches tentent de tirer partie de la régularité des champs de hauteurs pour en simplifier la représentation et en accélérer le rendu, tandis que les autres approches utilisent une forme de représentation non-régulière pour optimiser plus spécialement la qualité de rendu. On peut d'ores et déjà noter que les méthodes de visualisation basées sur des maillages réguliers (ou plus généralement semi-réguliers) [PG07] se montrent plus performantes et sont plus communes que les approches basées sur des maillages irréguliers.

Ainsi, on peut distinguer différentes classes de méthodes basées maillage : les maillages non-réguliers (section 2.2), les maillages semi-réguliers sans système de niveaux de détails continus (section 2.3) ou avec niveaux de détails continus (section 2.4). Parmi ces méthodes, il faut souligner les différences entre les méthodes à niveaux de détails continus et celles sans niveaux de détails continus. Dans le premier cas, le changement de niveaux de détails est effectué de façon invisible, au moyen de transitions douces entre les différentes géométries des différents niveaux tandis que dans le second cas, les changements sont effectués brusquement, ce qui peut se traduire par des apparitions soudaines de géométrie (*popping artefacts*).

Il est également intéressant de noter que les méthodes, décrites dans cette section, ont été développées en vue d'une exécution sur le processeur central, avant l'apparition des architectures graphiques programmables. L'apparition de ces processeurs a imposé un nouveau mode de fonctionnement pour les méthodes de visualisation de terrains et les méthodes CPU se sont révélées inadaptées à ces architectures et surtout inefficaces. Par conséquent, leur apparition a donné naissance à de nouvelles méthodes (décrites dans la section 3) qui sont basées, soit sur des regroupements de maillages (section 3.1), soit sur des grilles régulières imbriquées (section 3.2) ou persistantes (section 3.3).

2.1 Principe général

Toutes les différentes méthodes de visualisation, basées sur des maillages multi-résolutions, fonctionnent sur un schéma général similaire. On peut ainsi

en dégager différentes étapes clés qui sont, pour chacune des méthodes décrites dans cette section, traitées avec des approches différentes.

1. Extraction de la hiérarchie de niveaux de détails à partir du champ de hauteurs représentant les données initiales. Lors de cette étape, la représentation hiérarchique des données est construite en définissant, généralement, un ensemble de maillages de résolutions différentes et décroissantes.
2. Sélection du niveau de détail. Pendant le rendu, un parcours de la hiérarchie est effectué afin de sélectionner le (ou les) niveau(x) de détails à rendre. Cette sélection est effectuée suivant une évaluation de la qualité du rendu calculée via une heuristique ayant recours à une métrique d'erreurs.
3. Construction - extraction du maillage à rendre. À partir du résultat du parcours de la hiérarchie de l'étape 2, un maillage représentant le terrain est construit. Il peut être généré de façon à intégrer des transitions continues entre différents niveaux de détails.
4. Rendu. Le maillage est envoyé à la carte graphique pour rastérisation et illumination.

L'étape 1 est généralement effectuée lors d'une phase pré-traitements. Par contre, les étapes 2, 3 et 4 sont effectuées à chaque image rendue. Ce sont ces étapes de construction de maillage, à chaque image, qui empêchent une exécution efficace de ces méthodes sur les processeurs graphiques actuels. En effet, elles créent des transferts continus entre la mémoire centrale et la mémoire graphique, sur des masses de données importantes ce qui impacte directement sur les performances de rendu.

2.2 Maillages non-réguliers

Les maillages triangulaires irréguliers (*triangulated irregular networks*, *TIN*) ont été introduits dès la fin des années 70 par Peucker, Fowler *et al.* [PFLM78, FL79]. Ces triangulations sont extraites du champs de hauteurs mais ne conservent pas l'organisation régulière (sur une grille) des sommets. Leur intérêt majeur est qu'elles permettent d'approximer finement la surface du terrain, tout en minimisant le nombre de triangles générés et affichés.

En utilisant une hiérarchie de ces triangulations, à différents niveaux de précision, il est possible de créer un mécanisme de niveaux de détails continus. Pour extraire de telles hiérarchies de triangulations, il existe plusieurs méthodes : via des triangulations de Delaunay [COL96, CPS97] ou en utilisant des maillages de connectivités arbitraires, comme l'extension des *progressive meshes* dédiée aux terrains et développée par Hoppe [Hop98].

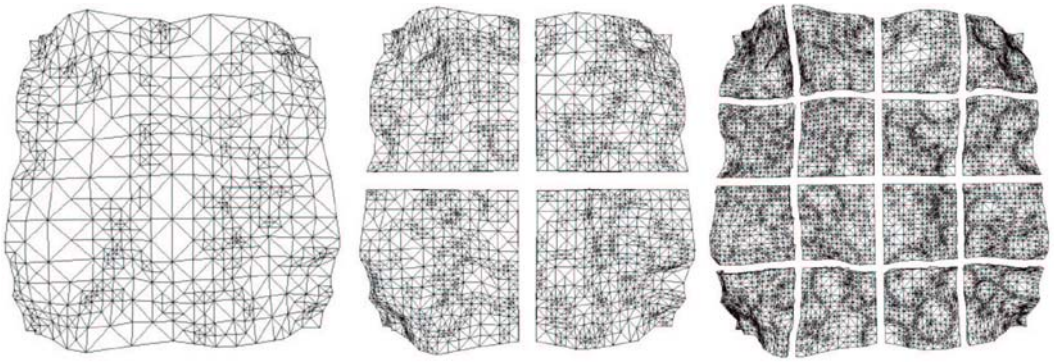


Figure 6.1 – Hiérarchie de niveaux de détails dans le cas d'une division par tuiles de terrain. Ici les trois premiers niveaux de détails d'une hiérarchie d'arbre quaternaire sont présentés. Extrait de [Ulr00].

Ces représentations irrégulières permettent le meilleur rapport entre le nombre de triangles et la qualité de la représentation du terrain. Cependant, elles ont généralement recours à des structures de données complexes à mettre en œuvre. Par conséquent, ces méthodes utilisent intensivement le processeur central et se trouvent inadaptées aux processeurs graphiques actuels. Néanmoins, ce type de représentation peut être combiné avec des structures s'appuyant sur la régularité des champs de hauteurs, notamment dans le cadre des méthodes basées sur des regroupements de triangles.

2.3 Maillages semi-réguliers sans niveaux de détails continus

Ces méthodes sont basées sur une approche tirant parti de la régularité des données. Elles proposent une hiérarchie statique de maillages représentant le terrain à différentes résolutions. Une des stratégies permettant de la générer, repose sur une division du terrains en cellules (généralement appelées *tuiles*). À chaque tuile est associée une hiérarchie de maillages, où chaque niveau est constitué par un maillage triangulaire construit pour une résolution fixe, comme l'illustre la figure 6.1 dans le cas d'une division suivant un arbre quaternaire.

Le niveau le plus fin est, le plus souvent, constitué par un maillage à la résolution des données. Chaque niveau de détails suivant est construit en simplifiant le maillage du niveau précédent. Les schémas de division et de subdivision employés suivent généralement le principe de division par arbres quaternaires (*quad-trees*) [RLIB99]. Dans le cas d'une division simple, comme celle proposée par Falby *et al.* [FZPM93], chaque niveau de détails est représenté par un maillage construit sur un nombre fixe d'échantillons. Le choix du niveau de détails à utiliser, au moment du rendu, pour une tuile s'effectue

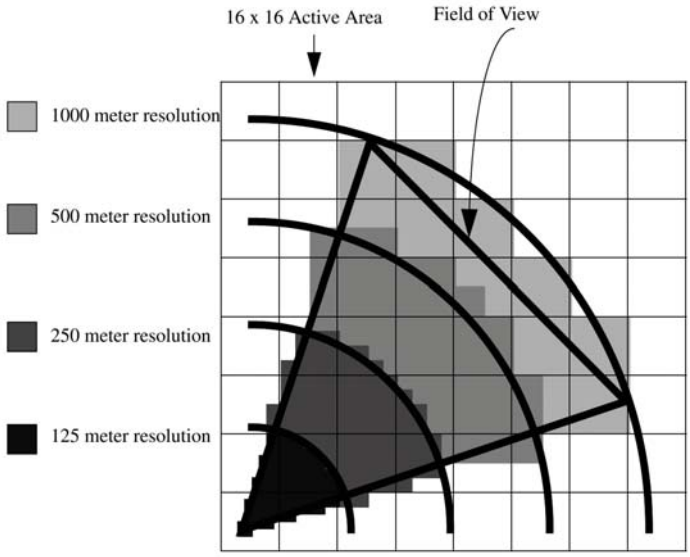
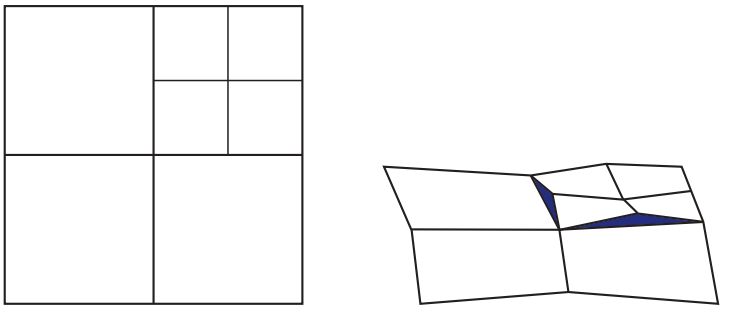


Figure 6.2 – Choix du niveau de détails dans la méthode de Falby *et al.* [FZPM93]. Le choix du niveau de détails de chaque tuile de terrain à utiliser se fait uniquement en se basant sur la distance entre la caméra et la tuile de terrain concernée. Extrait de [FZPM93].

simplement en fonction de la distance entre le point de vue et la tuile, comme l'illustre la figure 6.2.

Ces méthodes se montrent relativement simples à implanter et sont tout à fait adaptées à la gestion de très grandes masses de données, sous réserve de disposer d'un mécanisme efficace de chargement des tuiles. Néanmoins, leur principal problème réside dans la gestion des transitions entre les différents niveaux de détails. En effet, les maillages représentant deux tuiles adjacentes étant totalement indépendants, il est possible que des fissures se forment aux jointures des différentes tuiles, comme illustré par la figure 6.3. Pour remédier à ce problème, plusieurs techniques ont été proposées, comme un chevauchement des maillages représentant deux tuiles adjacentes. Des techniques plus complexes, introduisant, au moment du rendu, des morceaux de maillage pour assurer les jonctions entre les différentes tuiles, ont également été développées [Ulr00]. Ces ajouts compliquent malheureusement l'implantation, pourtant initialement simple, de ces techniques et nécessitent des traitements effectués sur le processeur central, durant l'étape de rendu et cela pour chaque image. Par conséquent, ces méthodes peuvent difficilement être portées afin de les exécuter sur les processeurs graphiques actuels. Elles sont donc peu efficaces en terme de performances de rendu.

De plus, le passage d'un niveau de détails à un autre est instantané : aucun mécanisme n'assure des transitions continues et douces entre les différents



(a) Maillage de quadrilatères basée sur une division par un arbre quaternaire.

(b) Fissures aux jointures entre différents niveaux de détails.

Figure 6.3 – Exemple de fissures formées aux jointures entre différents niveaux de détails.

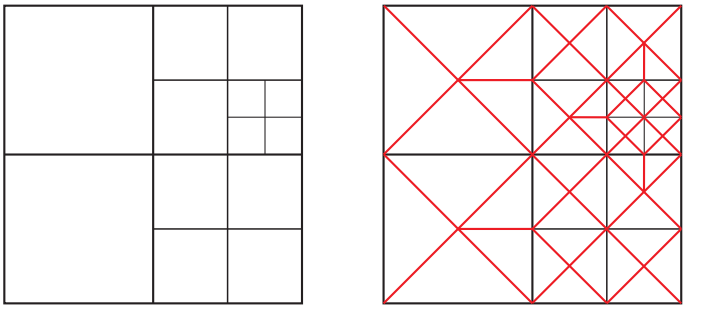
niveaux de détails. Par conséquent, le passage d'un niveau de détails à l'autre est brusque et se traduit par des effets de *popping*.

Au titre des extensions de ces méthodes basées sur une division régulière des données, on peut noter les travaux de Baumann *et al.* [BDHK99] qui proposent d'utiliser une division régulière mais offrent la possibilité d'utiliser différents type de représentations pour certains niveaux de détails (maillages uniformes, non uniformes ou TINs). Cette approche hybride est également reprise dans le cadres des méthodes de visualisation de terrains basées sur des *clusters* de triangulations.

2.4 Maillages semi-réguliers avec niveaux de détails continus

À l'opposé des méthodes basées sur des maillages semi-réguliers ne proposant pas de mécanisme de niveaux de détails continus, les méthodes qui en fournissent un, effectuent des transitions douces entre les différents niveaux de détails. Les transitions sont donc progressives ce qui tend à réduire, voire à supprimer, les effets de *popping*. Pour cela, ces méthodes ne considèrent plus le terrain suivant une division en tuiles mais le traitent dans sa globalité. Cela permet de corriger plus facilement les problèmes (et en particulier les fissures) qui surviennent aux jonctions entre différents niveaux de détails, mais compliquent leur implantation à cause, notamment, de la gestion de la masse de données. Ces méthodes sont très populaires et ont donné lieu à de nombreux travaux présentés en détails dans l'état de l'art proposé par Pajarola et Gobetti [PG07].

Pour générer la structure hiérarchique de niveaux de détails, différentes stratégies sont utilisées mais reposent toutes sur une représentation par des



(a) Division suivant une hiérarchie d'arbres quaternaires.

(b) Exemple de triangulation résultante [SS92].

Figure 6.4 – Exemples de triangulations basées sur une division suivant un arbre quaternaire : (a) hiérarchie de base et sa triangulation correspondante (b) (en rouge).

arbres quaternaires ou des arbres binaires. La construction de ces hiérarchies est généralement effectuée, soit par simplification (regroupement de paires de triangles), soit par raffinement (division d'un triangle en deux triangles de taille plus réduite). Ces différentes étapes de création de la structure hiérarchique peuvent être encodées de manière compacte, ce qui permet de représenter la hiérarchie de maillages avec une occupation mémoire réduite.

Comme mentionné lors de la description du principe général de ces méthode, à la section 2.1, les étapes clés de ces méthodes sont les étapes de parcours de la hiérarchie et de génération du maillage. Pour cela, des algorithmes efficaces pour traverser la hiérarchie et pour remailler le terrain ont été proposés. Parmi ces différentes et très nombreuses méthodes, on peut citer les travaux menés par Lindstrom *et al.* [LKR+96, LP01] ainsi que ceux de Pajarola [Paj98].

Les champs de hauteurs se prêtent assez bien à la représentation d'une hiérarchie via un arbre quaternaire [SS92]. Cette hiérarchie est construite suivant certaines règles, afin de garder la hiérarchie de maillages conforme, ce qui permet de limiter l'introduction de fissures entre les différents niveaux de détails lors de la triangulation. Une hiérarchie de maillages est ainsi dite conforme lorsque les niveaux de détails de régions adjacentes ne peuvent différer au plus de un. Un exemple de ce type de représentation est donné par la figure 6.4a. À partir de cette hiérarchie, est construite une triangulation qui représente le terrain, comme l'illustre la figure 6.4b. Plusieurs stratégies existent pour bâtir cette triangulation. On peut citer celle de Lindstrom *et al.* [LKR+96] reprise et étendue par Pajarola [Paj98].

Les maillages dits *4-8* (*cf.* figure 6.5) forment un autre type de subdi-

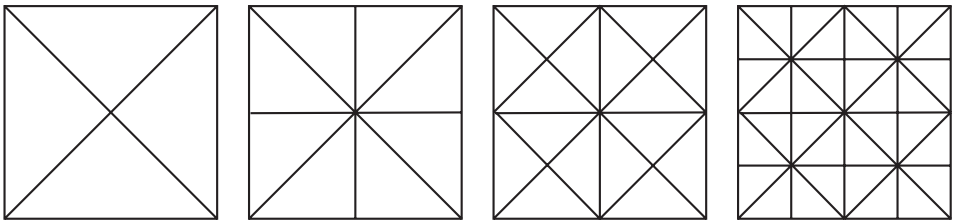


Figure 6.5 – Exemple de subdivision de maillages 4-8 : tous les sommets (hors bord) composant ces maillages ont une valence de 4 ou de 8.

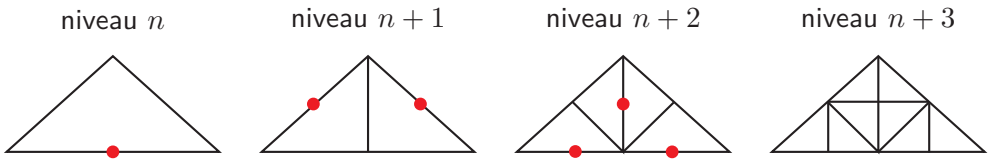


Figure 6.6 – Division binaire de triangles suivant l'arête la plus longue. En rouge, les sommets insérés pour former le maillage du niveau de détails suivant.

visions également employé pour mailler les hiérarchies d'arbres quaternaires, comme présentés, entre autres, par Balmelli *et al.* [BAV98]. Cette technique utilise la notion de domaine de fusion qui permet de marquer les sommets à regrouper lors de la simplification. Un principe symétrique est utilisé pour gérer l'ajout de sommets dans le maillage pour subdiviser celui-ci.

Afin de bénéficier des avantages des maillages irréguliers qui se montrent plus fidèles dans l'approximation faite de la surface du terrain, tout en tirant parti de la régularité des données, Pajarola *et al.* [PAL02] introduisent les *QuadTIN*. Ils proposent une structure de données permettant de combiner une décomposition en arbres quaternaires et une représentation par des maillages irréguliers. Dans ce cas, lors de la génération du maillage, les sommets ne sont pas forcément insérés au milieu des arêtes mais peuvent être légèrement translatés pour augmenter la qualité de l'approximation de la surface du terrain, toujours en fonction d'une métrique d'erreur.

Les méthodes, utilisant une représentation hiérarchique via des arbres binaires de triangles, ont généralement recours à une division des triangles suivant l'arête la plus longue, comme le décrit la figure 6.6. Cette approche permet, en particulier, de conserver un maillage régulier. Parmi ces méthodes, le *real-time optimally adapting meshes* (*ROAMing*), introduit par Duchaineau *et al.* [DWS+97] définit un système basé sur une double file de priorités indiquant les triangles à subdiviser ou à fusionner. Le choix des triangles à traiter s'effectue en s'appuyant sur une métrique d'erreur particulière. Ce système, très complet, définit également certaines extensions permettant d'augmenter

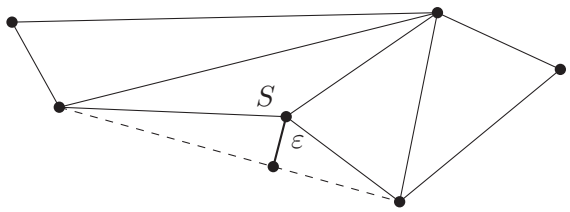


Figure 6.7 – Exemple de mesure d'erreur ε pour un sommet S donné. Cela revient à calculer la distance entre ce sommet et la surface du terrain sans ce sommet.

les performances générales, notamment par une optimisation progressive du rendu, une technique efficace de *frustum culling* (permettant de ne pas rendre les zones non visibles) ou encore une méthode permettant de créer les bandes de triangles à rendre, suivant un ordre optimal pour la rastérisation. Evans *et al.* [EKT01] proposent encore certaines améliorations au *ROAMing*, et en particulier une structure de données efficace pour stocker la hiérarchie de triangles.

Ces différentes méthodes ont été très populaires grâce à de très bonnes performances, aussi bien en terme de qualité que de vitesse sur les matériels graphiques pour lesquels elles ont été développées. Cependant, leur implantation nécessite de reconstruire, à chaque image rendue, un maillage complet du terrain. Cette contrainte, très forte, leur interdit une exploitation efficace des architectures graphiques actuelles. Leur implantation est donc restreinte au processeur central, ce qui ne leur permet pas de rivaliser en terme de performances de rendu avec les méthodes exploitant totalement les architectures matérielles actuelles.

2.5 Métriques d'erreur

Afin d'extraire un maillage représentant le terrain, lors du parcours de la hiérarchie de niveaux de détails, les méthodes présentées précédemment doivent évaluer la qualité de l'approximation de la surface du terrain. Pour cela, elles s'appuient, pour la plupart, sur une métrique d'erreur : pour chaque image, le maillage généré et rendu doit minimiser cette erreur. En fonction du type de méthode, ces métriques d'erreurs peuvent être définies soit en espace objet, soit en espace image. Leur stockage peut également varier : une valeur d'erreur peut-être associée à chaque sommet ou à chaque triangle ou encore être associée à des regroupements de triangles.

D'une façon générale, ces métriques évaluent une distance verticale entre la surface réelle du terrain et son approximation à un niveau de détail donné

ou la différence entre deux niveaux de détails. Concrètement cela se traduit en mesurant la distance verticale, pour un sommet donné, à un niveau de détail donné, et la même surface mais en ayant retiré le sommet, comme illustré par la figure 6.7. Le calcul de la distance peut être effectué soit en espace objet, comme Pajarola [Paj98], ou bien en espace image, comme par exemple Lindstrom *et al.* [LKR+96] ou encore Duchaineau *et al.* [DWS+97]. Il est généralement effectué pendant la phase de construction de la structure hiérarchique.

Les métriques d'erreurs définies en espace objet et associées aux sommets sont généralement utilisées dans le cas des hiérarchies basées sur une approche par arbres quaternaires [SS92]. Il est à noter que ce type de métriques permet d'encoder la structure hiérarchique [Paj98] et est assez simple à stocker (un attribut supplémentaire est simplement ajouté aux sommets). Toutefois, ces métriques définies en espace objet souffrent de certaines limitations. En effet, à titre d'exemple, aucune prise en compte de la projection de visualisation n'est intégrée dans leur évaluation. Par conséquent il est difficile, pour les mécanismes de niveaux de détails, de les moduler avec un critère précis de distance au point de vue, qui permet, entre autres, de dégrader la qualité du rendu des zones lointaines du terrain, qui sont donc peu visibles, afin d'augmenter la vitesse de rendu. C'est pourquoi il est parfois préférable d'utiliser des métriques d'erreurs basées images.

Ces dernières permettent une adaptation de la représentation du terrain aux conditions de visualisation, comme l'illustre Lindstrom *et al.* [LKR+96]. Ce type de métriques, associant une valeur d'erreur pour chaque triangle, est fréquemment utilisée dans le cadre des techniques basées sur des arbres binaires de triangles, comme pour le *ROAMing* [DWS+97]. Il a également été montré [LP01, LP02] que ces métriques peuvent-être saturées, c'est-à-dire que l'erreur d'un triangle peut être propagée aux triangles des niveaux de détails sous-jacents, ce qui permet d'encoder facilement les dépendances entre les triangles. Bien que plus efficace dans la pratique, ce type d'approche pour les métriques d'erreurs est plus délicat à mettre en œuvre. Son stockage est, en effet, plus complexe qu'une simple association entre une valeur et un sommet. De plus, comme il faut stocker une valeur d'erreur pour chaque triangle de la hiérarchie, cela représente plus de valeurs à stocker que dans le cas d'une hiérarchie basée sur des arbres quaternaires (une valeur d'erreur par sommet) et par conséquent nécessite plus de mémoire. Afin de réduire encore la quantité de valeurs à stocker pour représenter l'erreur, elles peuvent être associées à des regroupements de triangles. C'est le cas des métriques employées pour les méthodes basées sur des regroupements de triangles ou de maillages.

3 Évolution GPU des méthodes de visualisation de terrains basées maillages

La plupart des méthodes présentées jusqu'à présent ont été développées pour une exécution sur CPU, avant l'apparition des processeurs graphiques programmables. Par conséquent, elles ne permettent pas d'exploiter pleinement les capacités offertes par ce type d'architectures matérielles : elles ont fréquemment recours au mode de rendu immédiat qui impose un transfert constant d'informations entre la mémoire centrale et la mémoire graphique. Afin de tirer parti de la puissance de calculs offerte par ces nouveaux processeurs graphiques, des méthodes de visualisation de terrains ont été développées en s'appuyant notamment sur des regroupements de triangles en *clusters* (section 3.1), ou via l'utilisation de grilles régulières imbriquées (section 3.2) ou persistentes (section 3.3).

3.1 Visualisation de terrains par regroupements de triangulations

Ces méthodes fonctionnent sur le principe suivant : la géométrie représentant le terrain est découpée en différents groupes de triangulations (*clusters*). Ce découpage est hiérarchique et suit un schéma similaire à un maillage multi-résolutions semi-régulier (en utilisant des découpages suivant des arbres binaires de triangles ou des arbres quaternaires). Chaque *cluster* représente une triangulation d'une partie du terrain, à différents niveaux de détails. Ces *clusters* sont généralement pré-calculés puis peuvent être mis en cache directement au niveau de la mémoire graphique ce qui permet de réduire grandement les échanges entre la mémoire centrale et la mémoire graphique qui pénalisent très fortement la vitesse de rendu.

Un premier type d'approches [Wag03, WMD+04, SW06] reprend le concept des méthodes basées sur des maillages semi-réguliers sans mécanisme de niveaux de détails continus, c'est-à-dire en divisant le terrain en tuiles fixes, telles que décrites à la section 2.3. Lors d'une étape de pré-traitements, le terrain est divisé en différentes tuiles, pour lesquelles est construite une hiérarchie de maillage. Chaque niveau de chaque hiérarchie représente donc un maillage qui peut être mis en mémoire cache dans la mémoire graphique. Les niveaux de détails sont généralement choisis indépendemment pour chaque tuile et sont rendus indépendemment les uns des autres. Il en résulte des problèmes de fissures au niveau des jointures entre les différentes tuiles qui nécessitent des traitements complexes (génération de maillages à la volée aux bordures, ...) pour être corrigés.

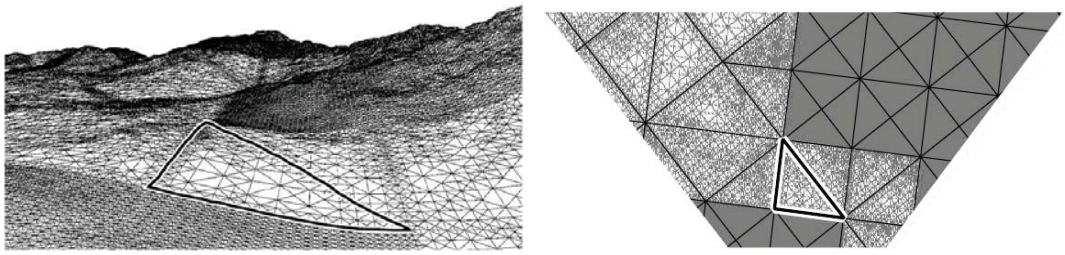


Figure 6.8 – Regroupement de triangles dans *CABTT*. Vue en fil de fer d'un terrain rendu avec en gras la matérialisation d'un regroupement de triangles. Extrait de [Lev02].

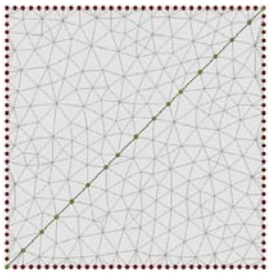


Figure 6.9 – Exemple du modèle hiérarchique employé dans *BDAM* [CGG+03a]. La structure régulière est bâtie sur une hiérarchie de triangles rectangles. Chaque triangle de cette hiérarchie représente un maillage irrégulier de triangles. Extrait de [PG07].

Un autre type d'approches consiste à adapter au GPU des méthodes existantes. C'est le cas des méthodes *RUSTiC* (*ROAM Using Surface Triangle Cluster*), développée par Pomeranz [Pom00], et *CABTT* (*Cached Aggregated Binary Triangle Trees*), introduite par Levenberg [Lev02], qui adaptent toutes les deux l'algorithme *ROAM* proposé par Duchaineau *et al.* [DWS+97]. Dans ces adaptations, les triangles formant la subdivision d'une région (triangulaire) du terrain sont regroupés dans un *cluster* de triangles qui peut lui même contenir plusieurs *clusters* de sa subdivision formant les niveaux de détails plus fins, comme illustré pour *CABTT* par la figure 6.8. De cette façon, les *clusters* de triangles peuvent être mis en cache sur le GPU après avoir été générés lors d'une phase de pré-traitements, dans le cas de *RUSTiC*, ou à la volée, dans le cas de *CABTT*. Finalement, le choix du niveau de détails se fait au niveau des *clusters* de triangulations et non plus simplement par triangles comme dans l'algorithme *ROAM* original.

La dernière approche retenue pour ces méthodes de regroupement en *cluster* de triangles propose de découper les maillages en combinant des structures hiérarchiques régulières avec des maillages irréguliers. Dans le cas de *BDAM* (*Batched Dynamic Adaptive Meshes*) et de son extension *P-BDAM*

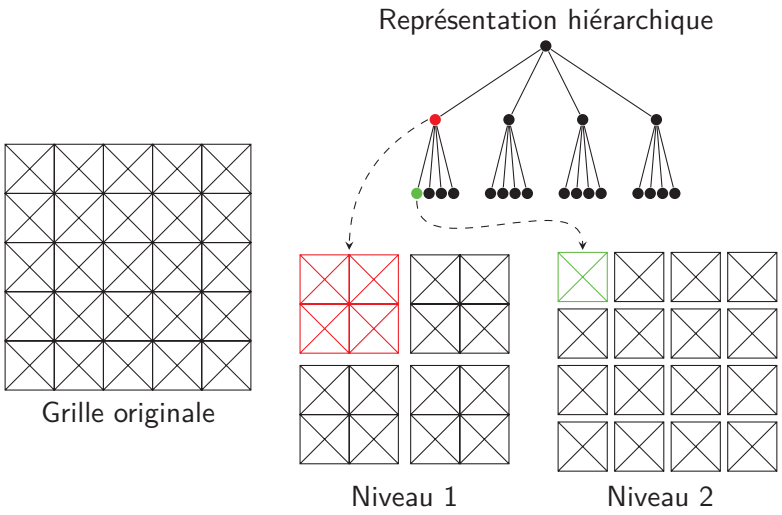


Figure 6.10 – Hiérarchie et subdivision dans le cadre des *QuadTIN*. Le terrain est subdivisé hiérarchiquement en suivant un arbre quaternaire. Chaque noeud de l'arbre correspond à un bloc de maillage irrégulier.

(*Planet-sized Batched Dynamic Adaptive Meshes*), toutes les deux proposées par Cignogni *et al.* [CGG+03a, CGG+03b], une structure hiérarchique, régulière et adaptative est construite pour gérer le mécanisme de niveaux de détails. Sa construction s'appuie sur une hiérarchie de triangles rectangles. Au final, un *cluster* de triangles est associé à chaque triangle rectangle de la hiérarchie. Les différents *clusters* sont formés par des maillages irréguliers de triangles, comme l'illustre la figure 6.9. Le choix du niveau de détails est effectué en se référant à une métrique qui associe à chaque *cluster* une valeur d'erreur. Dans le même esprit, Lario *et al.* [LPT03] proposent une extension des *QuadTIN* (*Quadtree-based Triangulated Irregular Networks*) [PAL02] également basée sur une structure hiérarchique définie par un arbre quaternaire où chaque noeud de l'arbre représente un bloc de maillages irréguliers. Cette hiérarchie est illustrée par la figure 6.10. Ce principe est également utilisé par Hwa *et al.* [HDJ05] mais dans le cadre de triangulations basées sur des maillages 4-8. Cette technique est particulièrement bien optimisée pour gérer des grilles régulières. Plus récemment, Bösch *et al.* [BGP09] proposent une technique permettant d'accélérer le mécanisme de sélection des niveaux de détails à employer. Leur structure hiérarchique est définie en combinant une subdivision par une hiérarchie d'arbres binaires de triangles (utilisée par le mécanisme de niveaux de détails) et une subdivision par arbres quaternaires (utilisée pour représenter les données).

D'une façon générale, ces méthodes génèrent un plus grand nombre de triangles à rastériser que les méthodes CPU mais comme elles bénéficient de la pleine accélération matérielle du GPU, elles obtiennent des performances de rendu bien plus élevées. Par conséquent ces méthodes ont pris le pas sur

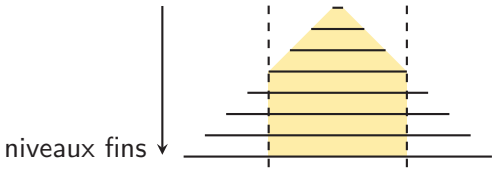


Figure 6.11 – Principe des *clipmaps* de Tanner *et al.* [TMJ98]. La région jaune matérialise la zone de la pyramide de *mipmaps* mise en cache dans la mémoire graphique.

les méthodes hiérarchiques développées pour le CPU. À l'heure actuelle, elles représentent la majorité des techniques de visualisation de terrains et sont de plus en plus fréquemment couplées à des techniques de compression de la géométrie [GMC+06, DSW09] afin d'accroître encore la taille des terrains pouvant être visualisés.

3.2 Grilles régulières imbriquées

La méthode décrite dans cette sous-section peut également être rattachée aux méthodes basées sur des maillages réguliers non-adaptatifs, telles que présentées dans la section 2.3 mais son approche particulière et ses performances globales de rendu, nous ont poussé à la présenter de façon plus détaillée.

Les *geometry clipmaps* ont été initialement présentées par Losasso et Hoppe [LH04] avant d'être entièrement portées pour une exécution sur les processeurs graphiques par Asirvatham et Hoppe [AH05]. Cette méthode est basée sur le principe des *clipmaps*, introduites par Tanner *et al.* [TMJ98], qui proposent un modèle permettant de représenter dynamiquement une pyramide de *mipmaps* [Wil83] de taille arbitraire mais occupant un espace un mémoire fixe, pré-défini à l'avance. La figure 6.11 illustre ce mécanisme : seule une partie de la pyramide de textures composant les *mipmaps* est mise en mémoire graphique, le reste des données étant chargé dynamiquement en fonction des besoins.

Les *geometry clipmaps* appliquent exactement le même principe à la géométrie d'un terrain. Son champ de hauteurs correspondant est stocké dans une texture qui est utilisée pour construire une pyramide de *mipmaps*. Cette pyramide permet ainsi de générer la hiérarchie de niveaux de détails utilisée pour représenter le terrain. Afin de gérer des champs de hauteurs de taille arbitraire, une *clipmap* est utilisée pour l'accès aux données. Pour maintenir une cohérance entre le chargement des données et la visualisation du terrain, la position de la caméra et le centre de la pyramide évoluent en parallèle de façon à maintenir la caméra au centre de la pyramide. Par conséquent, les

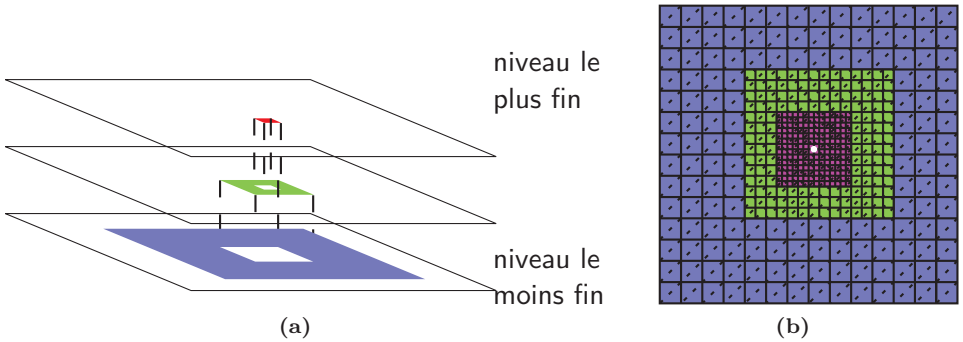


Figure 6.12 – Schéma de principe des *geometry clipmaps*. Exemple pour trois niveaux de détails. (a) : vue de la pyramide de champs de hauteurs. Chaque partie colorée correspond à la partie de terrain rendue suivant la grille de maillage vue sur la figure (b). Le point blanc correspond à la position de l'observateur.

données formant la *clipmap* sont mises à jour en fonction des déplacements de la caméra.

Le rendu est effectué via un ensemble de grilles régulières imbriquées les unes dans les autres dont la résolution décroît à mesure que l'on s'éloigne du centre de la pyramide (*cf.* figure 6.12). Chaque grille correspond à un niveau de détails de la *clipmap*. Comme la hiérarchie est construite en suivant le principe des *mipmaps*, la résolution de la grille de niveau l correspond simplement à la résolution de la grille de niveau $l-1$ divisée par deux (*cf.* figure 6.12b). Afin d'éviter des transitions brusques entre les différents niveaux de détails, une composition (*blending*) géométrique est appliquée sur une zone de transitions entre les différents niveaux de détails : la géométrie du niveau le plus fin est déplacée pour la faire tendre vers la géométrie du niveau le moins fin.

Les *geometry clipmaps* proposent également une stratégie efficace et incrémentale de chargement des données en s'appuyant sur un schéma toroïdal. La structure de grilles régulières imbriquées, couplée à la représentation des données dans une *clipmap*, permet de limiter grandement la quantité de données à charger et, par conséquent, les transferts entre la mémoire centrale et la mémoire graphique. De plus, ce mécanisme de chargement est couplé à un algorithme de compression et de décompression des données à la volée. Cette méthode, bien qu'effectuant des simplifications géométriques qui se traduisent par un lissage de la surface du terrain, et n'ayant qu'uniquement un système de niveaux de détails basé sur la distance au point de vue, bénéficie de performances de rendu de très haut niveau, aussi bien en terme de vitesse que de qualité. Ces performances peuvent en partie s'expliquer par une décroissance continue des détails en fonction de la distance qui se traduit par une taille

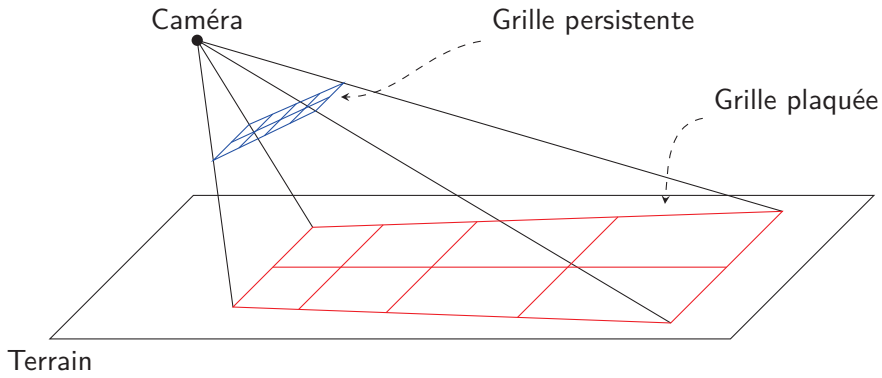


Figure 6.13 – Exemple de grilles persistantes. Le maillage est défini par une grille en espace image et est projeté sur le terrain.

quasi-constante des triangles projetés à l'écran et ceux, pour tous les niveaux de détails. Cela permet une exploitation optimale de la chaîne de traitements du matériel graphique actuel.

3.3 Grilles persistantes

Dans ce type d'approche, le rendu s'effectue en employant une représentation du terrain en espace image. Très souvent, ces méthodes ont recours à une grille de maillage définie pour couvrir toute la fenêtre de vue. Cette grille est déformée pour s'adapter à la surface du terrain à visualiser. Différentes approches existent pour effectuer cette déformation. Dachsbacher et Stamminger [DS04] la déforment en fonction d'une *geometry image* [GGH02] construite à partir du champ de hauteurs et des paramètres de visualisation. Livny *et al.* [LSGS08], se basant sur les travaux de Johanson [Joh04] pour le rendu de vagues d'océan, effectuent la déformation en projetant la grille sur le champ de hauteurs, comme le montre la figure 6.13. La projection est effectuée en lançant des rayons passant par chaque sommet de la grille. La position finale du sommet est ainsi donnée par l'intersection entre le rayon et le champ de hauteurs.

Ce type de méthodes souffre de problèmes liés à l'échantillonnage du terrain, effectué implicitement lors de la projection de la grille sur le champ de hauteurs. Elles offrent cependant la possibilité d'utiliser des grilles définies arbitrairement : aucune contrainte de régularité n'est posée sur ces dernières.

On peut noter qu'on retrouve ce même type d'approche dans le cadre de la visualisation de vagues d'océans, comme, par exemple dans les travaux de Hinsinger *et al.* [HNC02].

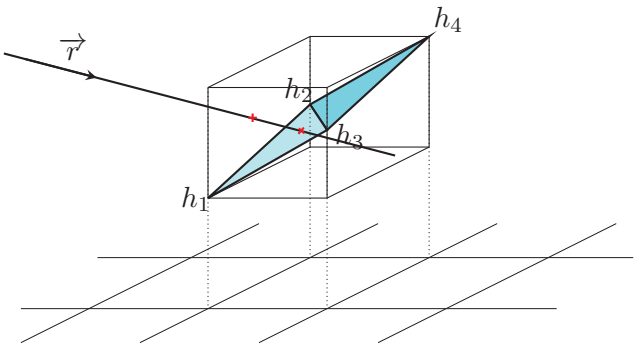


Figure 6.14 – Test d'intersection du lancer de rayons de Musgrave [Mus88]. Le test s'effectue entre le rayon $\overrightarrow{r}$ et deux triangles formés par les quatre élévations h_i du champ de hauteurs représentant la cellule courante.

4 Lancer de rayons

Les méthodes présentées jusqu'à présent s'appuient sur le mécanisme de rastérisation servant de base aux architectures matérielles graphiques. Cependant, l'apparition de processeurs graphiques programmables et offrant une grande puissance de calculs permet d'envisager une autre approche pour le rendu de terrains. En effet, aujourd'hui, il devient possible d'effectuer le rendu de terrains suivant des algorithmes qui fonctionnent en espace image qui exploitent en particulier la technique du lancer de rayons.

À l'origine des techniques de lancer de rayons adaptées aux champs de hauteurs, on trouve les travaux menés par Musgrave [Mus88]. L'objectif de ces travaux est de permettre le rendu de terrains, au moyen d'un algorithme de lancer de rayons qui utilise directement le champ de hauteurs représentant les données. Dans ces travaux, le lancer de rayons permet de générer un maillage qui est, par la suite, rastérisé en vue de son affichage. Le champ de hauteurs est interprété comme une grille dont chaque cellule est formée par les quatre élévations adjacentes du champ de hauteurs. Musgrave propose de parcourir les cellules suivant un rayon défini par le vecteur de vue et de tester, pour chaque cellule parcourue, si le rayon intersecte la surface du terrain ou non. Pour effectuer ce test, Musgrave propose de calculer le point d'intersection entre le rayon et les deux triangles formés à partir des quatre élévations composant la cellule, comme le montre la figure 6.14. Le parcours du rayon s'effectue, quant à lui, en le projetant sur le plan de base du terrain. Il est réalisé en utilisant un algorithme d'analyse de différences numériques (*digital difference analyzer, DDA*), semblable à l'algorithme de tracer de lignes de Bresenham qui permet de traverser les cellules formées par les élévations du champ de hauteurs. Au titre des extensions de cette méthode, on peut noter que Musgrave propose une

adaptation de cet algorithme afin d'extraire une représentation d'un terrain défini à partir d'un modélisation fractale [Mus94].

Afin d'améliorer les performances de ce type de méthodes, différentes optimisations ont été proposées, comme l'emploi de structures hiérarchiques ou de tests d'intersections différents [CS93, CORLS96]. Comme une accélération du parcours du rayon impliquerait une augmentation sensible des performances, il a été proposé des algorithmes pour l'optimiser [LS95, HS04] et, en particulier, des mécanismes permettant d'éviter les espaces vides (*empty space skipping*) [KRS05]. On peut aussi souligner que ces différentes techniques sont assez proches des méthodes de placage de déplacements utilisant également un lancer de rayons, comme celles décrites au chapitre 3.

Ainsi, dans la continuité des travaux de Musgrave, Henning et Stephenson [HS04] proposent de modifier l'algorithme permettant de projeter le rayon sur la grille de cellules du champs de hauteurs. Cette modification autorise l'implantation de ce type de techniques sur les processeurs graphiques actuels programmables. Kolb et Rezk-Salama [KRS05] présentent un mécanisme permettant d'éviter les espaces vides lors du parcours du rayon. Pour cela, ils définissent une zone de sécurité permettant de calculer une taille de pas variable, à utiliser lors du parcours, pour passer d'un point d'échantillonnage du rayon à un autre. Cette technique garantit qu'aucune intersection ne sera manquée lors de la progression. On peut également remarquer que cette méthode est également utilisée dans le cadre du placage de détails, comme décrit à la section 6 page 36 où elle y est présentée plus en détails.

Une approche différente pour éviter les espaces vides lors du parcours est présentée par Balciunas *et al.* [BDZ06]. Pour cela, ils proposent de combiner rastérisation et lancer de rayons. Lors d'une première passe de rendu, un maillage très simplifié du terrain, qui forme un volume englobant la surface réelle du terrain, est rendu dans un tampon de profondeur. Chaque pixel de ce tampon est ensuite utilisé pour servir de point de départ au lancer de rayons et permet ainsi de réduire la distance à parcourir sur chaque rayon. L'implantation proposée comporte quelques limitations : seule l'étape de génération du tampon de profondeur est effectuée par le processeur graphique, le lancer de rayons étant confié au processeur central avec toutes les conséquences que cela implique sur les performances.

En vue d'exploiter un algorithme de lancer de rayons dans le cadre d'une visualisation temps-réel des données, il est nécessaire d'utiliser intensivement le processeur graphique pour effectuer le parcours des rayons. En effet, il a été montré, en particulier par Purcell *et al.* [PBMH02], que les architectures graphiques programmables sont assez bien adaptées à ce type d'algorithmes.

Dans ce cadre Qu *et al.* [QQZ+03] proposent de modifier l'algorithme original de Musgrave [Mus88] pour qu'il puisse être exécuté sur GPU. Ils transforment ainsi la notion de cellules en voxels et remplacent le parcours de rayon *DDA* par un parcours rapide de voxels (*fast voxel traversal*), tel que présenté par Amanatides et Woo [AW87]. Dans ces travaux, l'accent est également mis sur la reconstruction de la surface, lors de la détection d'une intersection au sein d'un voxel, en remplaçant la paire de triangles par un morceau de surface bilinéaire.

Mantler et Jeschke [MJ06] proposent une approche dont l'objectif est plus spécialement la visualisation de terrains incluant de la végétation. Pour effectuer le rendu de terrain, ils utilisent un algorithme de lancer de rayons où le parcours est effectué en avançant le long du rayon et non plus en avançant sur le plan de base du champ de hauteurs, comme dans le cas de Musgrave. La végétation est représentée par un second champ de hauteurs, extrait à partir de la texture de couleurs du terrain, qui est superposé à celui du terrain lors de la phase de recherche d'intersections. Pour accélérer le parcours du rayon, la méthode proposée par Kolb et Rezk-Salama [KRS05] est employée.

Dick *et al.* [DKW09] proposent un système complet de visualisation de terrains comprenant notamment un mécanisme de compression et de chargement des données topographiques à la demande. Il supporte également la gestion de niveaux de détails qui a été développé précédemment par leur équipe [DSW09] pour un rendu basé sur des maillages triangulaires. Ce mécanisme de chargement de données fonctionne en utilisant un découpage du terrain en tuiles. Le rendu de chaque tuile s'effectue en exécutant un algorithme de lancer de rayons GPU, optimisé par l'utilisation d'un mécanisme de *maximum mipmaps*, tel que décrit par Tevs *et al.* [TIS08] et présenté précédemment (section 6). Il est intéressant de noter que dans cet article, les auteurs comparent et discutent les performances de rendu entre le lancer de rayons et la rastérisation, dans le cadre de leur mécanisme de chargement et de compression des données. Dans les cas où la rastérisation a recours à des triangles très petits (cas des jeux de données très denses et présentant des détails à hautes fréquences), le lancer de rayons devient plus avantageux que la rastérisation. La tendance s'inverse lorsque la taille des triangles devient plus importante. Cependant, ils émettent l'idée que la combinaison des deux techniques de rendu au sein d'une même méthode de visualisation peut s'avérer bénéfique.

5 Conclusion sur la visualisation de terrains

Jusqu'à présent, les méthodes de visualisation de terrains ont très largement été dominées par les méthodes mettant en place des structures basées sur des maillages hiérarchiques adaptatifs [LKR+96, DWS+97, Hop98, Paj98, LP01]. Avec l'apparition des processeurs graphiques programmables, ces méthodes se sont retrouvées totalement inadaptées à ce type d'architectures. Cependant, d'autres méthodes [Pom00, Lev02, PAL02, CGG+03a, LPT03, BGP09] inspirées sur ces principes de maillages multi-résolutions adaptatifs ont été développées en ayant recours à des mises en mémoire cache de morceaux de maillages. Finalement, ces méthodes offrent à l'heure actuelle un très bon compromis entre qualité de rendu et vitesse. Néanmoins, ces méthodes sont relativement compliquées à mettre en place et ont, le plus souvent, recours à des structures de données complexes nécessitant des étapes de pré-traitements pour les générer. La gestion de données dynamiques ou modifiables est donc difficilement envisageable avec cette catégorie de méthodes.

Les *geometry clipmaps* [LH04] proposent une approche différente mais qui permet une exploitation optimale du matériel graphique. Il en résulte une méthode extrêmement efficace et assez simple à mettre en place. Bien que de fortes approximations soient effectuées sur la géométrie rendue (notamment à cause d'un lissage inhérent à cette représentation des différents niveaux de détails, *cf.* chapitre 8), l'approximation de la surface est suffisante pour bon nombre d'applications nécessitant la visualisation de terrains. Cela en fait actuellement une des méthodes les plus efficace dans ce domaine, pour des applications ne nécessitant pas un rendu où l'erreur commise doit être précisément connue.

Longtemps inenvisageables pour des applications de visualisation temps réel de terrains, le lancer de rayons dédié aux champs de hauteurs représentant des terrains peut maintenant profiter de la puissance de calculs offerte par les processeurs graphiques actuels. Cela autorise leur utilisation sans trop sacrifier les performances, à l'exemple des différentes méthodes présentées dans la section 4 de ce chapitre [QQZ+03, MJ06, DKW09]. De plus, comme l'ont récemment montré Dick *et al.* [DKW09], un algorithme de lancer de rayons devient parfois plus intéressant qu'une méthode basée uniquement sur la rastérisation de triangles. Un des avantages non négligeable des algorithmes de lancer de rayons dédiés aux champs de hauteurs réside dans la possibilité qu'ont ces techniques à utiliser immédiatement des données sans avoir recours à des structures de données supplémentaires nécessitant habituellement de lourds pré-traitements. Par conséquent, aucune restriction n'est *a priori* posée en vue de leur utilisation sur des données modifiables de façon dynamique.

Chapitre 7

Visualisation de terrains : approche hybride

Dans ce chapitre nous présentons la méthode hybride que nous avons développée, avec pour objectif, la visualisation de données à deux dimensions et demie. Ce chapitre s'articule autour de différentes sections dont la première (section 1) présente les objectifs de cette méthode. S'en suit, à la section 2, un descriptif de la méthode présentant les choix techniques effectués pour développer la technique de visualisation. La section 3 présente une étude détaillée permettant l'évaluation de l'algorithme de lancer de rayons employé, et dont les résultats ont servi à établir le mécanisme adaptatif décrit à la section 4. Ce chapitre est conclu par quelques précisions d'ordre technique quant à l'implantation de la méthode sur le processeur graphique (section 5).

1 Présentation de l'approche

L'approche retenue pour cette méthode de visualisation de données à deux dimensions et demie consiste en une approche hybride, dans le même esprit que la méthode développée pour la visualisation de reliefs de faibles amplitudes. La justification du choix de cette approche est présentée dans cette section après un rappel de la problématique et des contraintes que nous nous sommes posées.

1.1 Problématique

Comme mentionné au chapitre présentant un état de l'art des principales méthodes de visualisation de terrains (*cf.* chapitre 6), le rendu de champs de

hauteurs a toujours été un sujet majeur en informatique graphique. Cependant, trouver un bon compromis entre vitesse de rendu, précision et flexibilité des méthodes de visualisation reste un problème complexe. En effet, beaucoup de domaines d'applications, comme les jeux vidéos, nécessitent une fréquence de rendu très élevée tandis que d'autres applications, comme la visualisation de données scientifiques ou les applications liées à la géomatique, ont besoin d'une très haute qualité de rendu, même si le taux de rafraichissement peut être plus faible. Dans certains cas, il est même nécessaire de pouvoir éditer dynamiquement ces données, durant leur visualisation. En effet, l'édition interactive de terrains pour la création de contenus graphiques dans le cadre de visualisations de paysages virtuels est un exemple majeur d'applications qui impose ce type de contraintes. Dans d'autres cas, l'aspect dynamique des données est également important tout au long, par exemple, du déroulement d'une simulation visant à reproduire l'érosion de terrain ou le mouvement de vagues d'océans, ou encore durant la visualisation de données scientifiques nécessitant l'édition d'une fonction de transfert, etc.

Comme nous l'avons présenté dans l'état de l'art, les méthodes existantes de visualisation de champs de hauteurs peuvent être classifiées en deux grandes catégories. La première regroupe les méthodes ayant recours à une structure hiérarchique de maillages, tandis que la seconde se compose de méthodes s'appuyant sur un algorithme de lancer de rayons, pouvant désormais être effectué sur le processeur graphique. Les méthodes, basées sur des maillages, bien que permettant d'obtenir d'excellentes performances, en termes de vitesse mais également en termes de qualité de rendu, souffrent de limitations assez fortes. En effet, ces différentes méthodes ont généralement recours à des étapes prétraitements assez lourdes pour générer leurs structures hiérarchiques et ne peuvent, de ce fait, pas utiliser directement les données brutes. À l'opposée, les méthodes utilisant un algorithme de lancer de rayons sur processeur graphique sont capables d' exploiter directement les données brutes. Dans ce cas l'exploitation directe des données est extrêmement intéressante car elle permet d'envisager des méthodes sans (ou seulement avec de très légers) pré-calculs qui, par conséquent, peuvent autoriser la visualisation de données variant au cours du temps, suite à un processus d'édition ou d'animation. Par conséquent, un de nos objectifs pour cette méthode de visualisation est la prise en compte de données pouvant varier au cours du temps. Pour cela, notre méthode de visualisation doit se montrer flexible et limiter au maximum les pré-traitements appliqués aux données, tout en évitant au maximum d'avoir recours à des structures de données complexes et tout en maintenant une vitesse de rendu interactive ou temps réel.

Cependant, et comme cela peut transparaître du chapitre 6, l'utilisation directe des données d'une carte d'élévations dans le cadre d'un lancer de rayons,

sans phase de pré-traitements, reste trop lente pour être satisfaisante dans la pratique. Quant à l'utilisation de techniques d'accélération, elle introduit malheureusement, soit des pré-traitements assez lourds, soit des approximations qui impliquent une précision moins élevée du rendu final. En effet, comme l'ont récemment montré Dick *et al.* [DKW09], le rendu de terrains, par un maillage, reste plus performant qu'un lancer de rayons pour des données dont la résolution est modérée (*i.e.* dont les détails ne sont pas trop fins). Cependant, ce type de rendu tend à introduire un lissage des données, comme cela peut être le cas avec les *geometry clipmaps* de Losasso et Hoppe [LH04]. Par conséquent, avec cette méthode, nous souhaitons également proposer une solution à ce lissage des données en développant une méthode de rendu hybride qui combine un rendu par maillage avec un lancer de rayons, et essayer d'en tirer, à chaque fois, le meilleur parti.

1.2 Approche hybride

Combiner deux méthodes de rendu, en utilisant des représentations des données différentes, n'a pas été, à notre connaissance, fortement exploré dans le domaine de la visualisation de terrains. En effet, seuls Dick *et al.* [DKW09] émettent juste la possibilité d'utiliser une telle approche pour visualiser un terrain, en combinant lancer de rayons et rastérisation de triangles. L'utilisation de ce type d'approche dans le cadre de l'ajout de détails, à base de champs de hauteurs, a été discutée dans la première partie de cette thèse. Dans le cadre plus général de la visualisation de données géométriques, on peut citer à titre d'exemple certaines méthodes qui ont été développées dans le but de combiner plusieurs types de représentations de données, afin d'augmenter les performances de visualisation. Parmi ces méthodes, on retrouve des travaux qui ont été menés pour combiner un rendu basé maillages et un rendu basé points [CN01, JLLW04]. Cependant, la majorité des travaux se concentre sur une combinaison de maillages à différentes résolutions, dans l'esprit des *progressive meshes* de Hoppe [Hop96]. Cela rapproche ces travaux des méthodes de visualisation de terrains, basées sur des maillages multi-résolutions, telles que celles décrites au chapitre 6. Dans une thématique plus proche du rendu réaliste, on peut mentionner des travaux, comme ceux de de Hakura et Snyder [HS01], liés au rendu des effets de réflexion et de réfraction utilisant un rendu hybride qui combine un rendu par des maillages avec un algorithme de lancer de rayons.

L'originalité de la méthode décrite dans ce chapitre réside donc dans l'approche hybride retenue pour effectuer le rendu des données. En combinant un rendu via un maillage et un algorithme de lancer de rayons, exécuté sur le processeur graphique, nous avons développé une méthode offrant un certain

degré de flexibilité tout en limitant les étapes de pré-traitements des données. Cela nous autorise à visualiser des données dynamiques, tout en maintenant un haut niveau de qualité, pour des performances de rendu qui demeurrent temps réel.

La méthode de visualisation, proposée dans ce chapitre, s'appuie donc sur une combinaison entre deux méthodes, sans avoir recours à un rendu par *bump mapping*. Cela s'explique, comme nous l'avons mentionné dans l'introduction de ce chapitre, par le fait que les reliefs visualisés, à l'échelle où sont généralement visualisées ces données, ne peuvent être rendu en les simulant. Leur amplitude est en effet trop importante et l'impact visuel de leur géométrie est donc trop forte pour être négligé. De plus, l'utilisation d'un lancer de rayons, en dégradant sa qualité permet de s'approcher des performances obtenues par le *bump mapping*, tout en conservant une meilleure information géométrique.

Pour cette technique, la représentation de la surface de l'objet, sous forme de maillage, est réservée aux zones des données nécessitant le niveau de précision le plus fin, tandis que le lancer de rayons est utilisé pour le reste du terrain. La précision de ce dernier est modulée afin de l'adapter aux reliefs formés par les données. Cela permet de trouver un équilibre entre la vitesse de rendu et sa qualité visuelle. La combinaison de ces deux méthodes s'appuie sur une heuristique basée sur une analyse de la qualité de rendu du lancer de rayons menée pour différentes conditions d'emplois et de visualisations. Les apports de ces travaux par les points suivants peuvent être ainsi résumés :

- une méthode hybride de rendu qui permet la combinaison entre un algorithme de lancer de rayons, appliqué pour chaque pixel de l'image, et un rendu basé sur des maillages, tous les deux exécutés entièrement sur le processeur graphique, ce qui permet à cette méthode d'atteindre des performances de rendu temps réel ;
- une heuristique simple, basée sur une évaluation de la qualité de rendu du lancer de rayons, permettant de combiner les deux méthodes de rendu retenues et de maintenir une qualité visuelle élevée ;
- une méthode n'ayant recours qu'à des structures de données simples, sans phase de pré-traitement lourde et complexe et qui permet, par conséquent, la prise en compte de données pouvant varier dynamiquement au cours du rendu.

2 Survol de la méthode

Le principe général de notre méthode de visualisation est décrit par le diagramme de la figure 7.1. Notre technique de visualisation s'appuie sur une

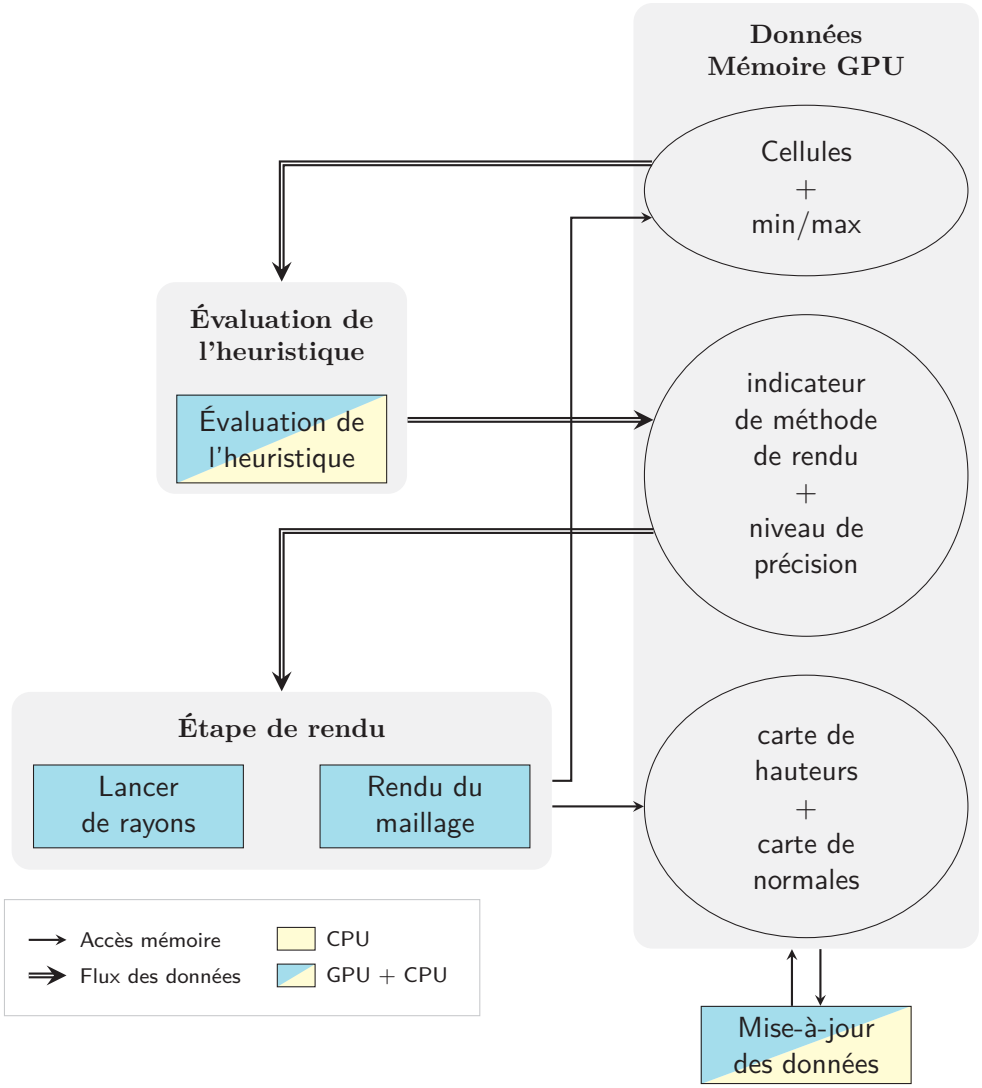


Figure 7.1 – Diagramme présentant le fonctionnement de notre méthode de visualisation. Les différentes flèches décrivent les flux de données entre chaque étape de traitements et la mémoire graphique.

décomposition cellulaire des données. Le champ de hauteurs est divisé en un ensemble régulier de cellules carrées, comme l'illustre la figure 7.2.

Au moment du rendu, pour chaque cellule, le mécanisme adaptatif, guidé par l'heuristique (*cf.* section 4), choisit le meilleur algorithme de rendu à utiliser entre le maillage et le lancer de rayons. Dans le cas où le lancer de rayons est sélectionné, le mécanisme adaptatif détermine également le niveau de précision avec lequel doit être effectué le rendu de la cellule courante. Les différents niveaux de précision sont prédéfinis et correspondent à des programmes de *shaders* particuliers. Afin d'améliorer les performances, ce mécanisme adaptatif

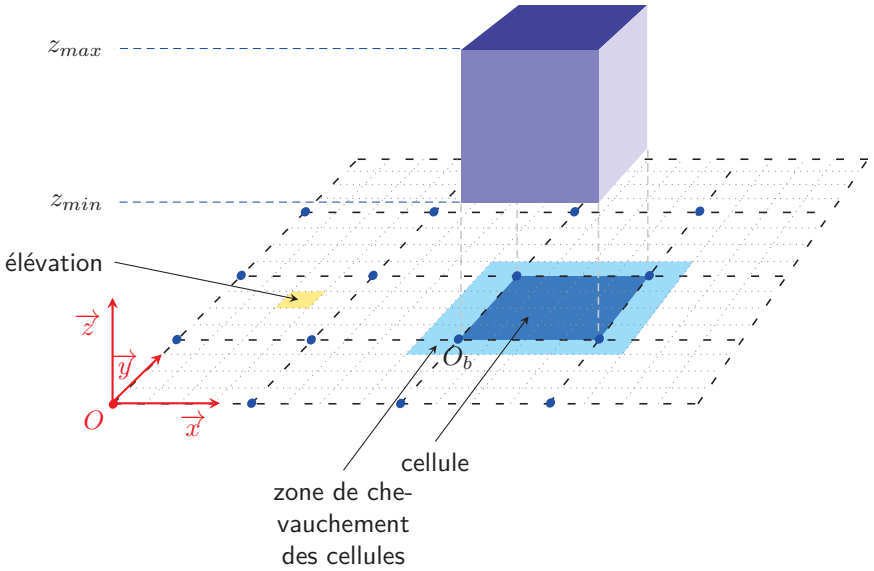


Figure 7.2 – Boîte englobant le relief d'une cellule.

intègre également un dispositif qui permet de supprimer les cellules non visibles lors du rendu d'une image. Le déroulement de ce mécanisme adaptatif peut ainsi être résumé par la série d'étapes suivantes :

1. l'heuristique est évaluée pour chaque cellule sur le processeur graphique (*cf.* sommet du diagramme de la figure 7.1) ;
2. la classification de l'heuristique est rapatriée de la mémoire graphique vers la mémoire centrale afin de trier, sur le processeur central, les différentes cellules suivant leur mode de visualisation et, le cas échéant, leur précision de rendu ;
3. les cellules sont rendues avec, en premier lieu, celles utilisant la rastérisation puis celles ayant recours au lancer de rayons (*cf.* bas de la figure 7.1).

On voit ainsi que la méthode de rendu est découpée en plusieurs parties qui s'articulent autour de l'heuristique, permettant le choix de la méthode de rendu la plus adéquate et de la précision de celle-ci.

2.1 Méthodes de rendu

En ce qui concerne le rendu via un maillage, nous avons opté pour un placage de déplacements par sommets. Quand une cellule est rendue via un maillage, un simple morceau carré de maillage est utilisé. Celui est construit pour contenir un nombre de sommets qui soit équivalent au nombre d'élévations qu'on retrouve dans une cellule. En d'autres termes, ce carré de maillage est de taille identique à une cellule et est défini afin que le placage de déplacements

puisse être effectué en associant chaque élévation de la cellule à un sommet de ce maillage.

Ce placage de déplacements est entièrement exécuté par le processeur graphique, sur un schéma semblable à l'algorithme décrit par l'algorithme 3.1 page 31. En ayant recours à cette technique de rendu, il devient possible de ne stocker qu'un seul maillage en mémoire graphique. Celui-ci est ensuite réutilisé pour chaque cellule rendue via la rastérisation.

Quand une cellule doit être rendu au moyen du lancer de rayons, nous utilisons, comme dans le cas de la visualisation des reliefs de faibles amplitudes, un algorithme dérivé du *relief mapping* de Policarpo *et al.* [POC05]. Pour cela, à chaque cellule est associée une boîte englobant son relief, comme l'illustre la figure 7.2. Les faces visibles de ces boîtes servent de support au lancer de rayons. Ainsi, pour chaque pixel rendu de ces boîtes, un rayon est lancé et l'intersection avec les données est calculée. La recherche d'intersections s'effectue via un parcours linéaire du rayon, couplé à une recherche dichotomique, comme cela est décrit par la figure 4.4 page 62. Chaque étape de recherche est effectuée avec un nombre d'itérations fixe. C'est sur ce paramètre que s'effectue la modulation de précision de l'algorithme : en réduisant ou en augmentant le nombre de pas de recherche effectué sur le rayon. On peut noter qu'il aurait été possible d'effectuer une modulation de précision par pixels directement au sein du *shader* implantant le lancer de rayons. Cependant, à cause des limitations matérielles imposées par les processeurs graphiques (notamment au niveau des branchements conditionnels), il est apparu plus performant de définir un ensemble de programmes de *shaders* où le nombre de pas de recherche est fixe. Ainsi, à chaque niveau de précision du lancer de rayons correspond une programme de *shaders* spécifique.

Afin de réduire la taille des boîtes, elles sont ajustées aux élévations contenues dans la cellule. En d'autre terme, ces boîtes sont construites en utilisant l'élévation minimale et l'élévation maximale de la cellule. Afin d'assurer la continuité entre les différentes boîtes englobantes, les altitudes maximales et minimales sont calculées en utilisant un pixel de chevauchement avec les cellules voisines (*cf.* figure 7.2). L'utilisation, pendant la construction des boîtes, de ces deux valeurs permet de réduire la taille des boîtes projetées à l'écran et, par conséquent, le nombre de rayons lancés pour chacune d'elles. Nous verrons également que ces boîtes englobantes jouent un rôle important au moment du choix du niveau de précision (*cf.* section 4).

2.2 Jeux de données dynamiques

Comme l'un des objectifs de cette technique de visualisation est la prise en compte des jeux de données pouvant varier au cours du temps, nous n'avons inclus aucun mécanisme d'optimisation à l'algorithme de lancer de rayons. Cela reste toutefois toujours possible. En effet si la contrainte de dynamisme des données n'est pas posée, il est tout à fait envisageable d'implanter un mécanisme d'accélération tels qu'un mécanisme basé sur les zones de sécurité [BD06, PO07] ou sur une structure hiérarchique [TIS08].

Comme cela est visible sur la droite du diagramme de la figure 7.1, la méthode de rendu proposée ne nécessitent qu'un nombre de données supplémentaires limité, en plus des données qui représentent la carte d'élévations :

- informations sur la division cellulaire des données (*i.e.* origine des cellules dans l'espace du champ de hauteurs, point O_b sur la figure 7.2) ;
- élévation minimale et maximale de chaque cellule (*i.e.* permettant de construire les boîtes englobant le relief des cellules).

Lorsque les données évoluent dynamiquement, les valeurs d'élévations minimale et maximale ainsi que les normales de la surface formée par le champ de hauteurs doivent également être mises-à-jour. Cependant, le calcul de ces différentes valeurs, pour chaque cellule, ne représente qu'une étape de calcul très légère qui peut-être effectuée sur le processeur graphique, en parallèle du calcul des normales. De plus, comme le terrain est divisé en un ensemble de cellules, lors d'une modification locale des données, il est possible de ne modifier que les cellules concernées, ce qui réduit fortement le coût de cette opération.

2.3 Structures de données

Comme la prise en charge des données dynamiques implique une exploitation directe de celles-ci, notre méthode ne s'appuie pas sur un mécanisme de niveaux de détails à l'échelle des données. En effet, utiliser une approche simple pour ce type de mécanisme, comme par exemple un *mipmaps* géométrique, ne fournit pas de contrôles suffisamment fins sur les simplifications géométriques opérées sur le données, comme cela est le cas sur le *geometry clipmaps*. Par conséquent, ce type de méthodes introduit un lissage excessif de la surface, comme nous le montrons au chapitre 8. Pour obtenir un meilleur contrôle, il faut employer une approche plus complexe pour le mécanisme de niveaux de détails qui doit se baser sur une analyse fine des données pour réduire le lissage. Cela implique obligatoirement des structures de données complexes à générer et à utiliser, au niveau du processeur graphique, ce qui impacterait directement les performances de rendu et rendrait quasiment impossible la visualisation de

données dynamiques.

Comme mentionné précédemment, sans la contrainte de dynamisme des données, le recours à de telles structures hiérarchiques est bien évidemment envisageable. Cependant, il faut toutefois noter que le gain de performances risque d'être minime. En effet, notre méthode est définie suivant un rendu en espace image pour les zones éloignées du point de vue, par conséquent, la complexité n'est pas directement liée à la taille des données [1] mais au nombre de pixels rendus. Néanmoins, les artéfacts liés à l'aliasing qui résultent d'une non correspondance entre la résolution des données et la résolution de l'écran seraient certainement réduits, voire éliminés, grâce au lissage de la surface du champ de hauteurs introduit par ces mécanismes.

3 Étude comparative entre rendu par lancer de rayons et rendu par maillage à pleine résolution des données

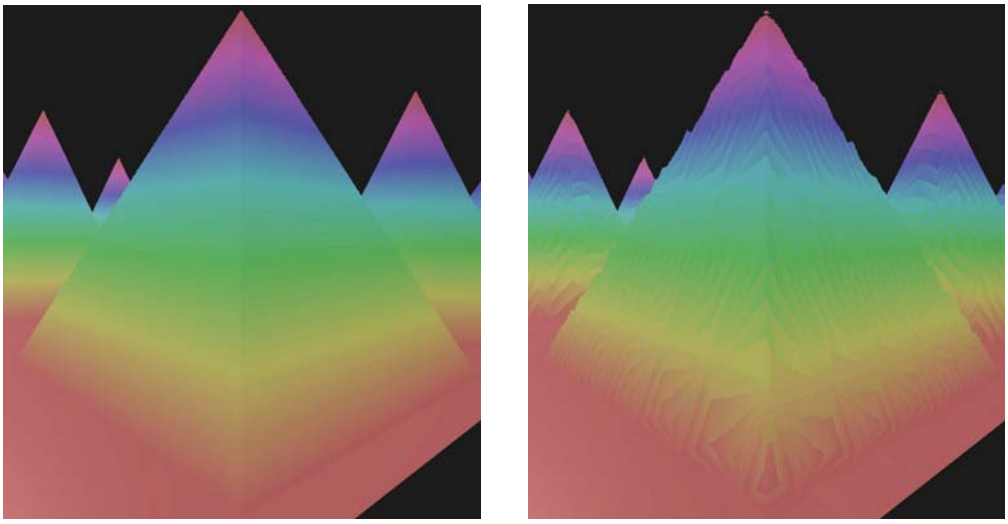
Comme décrit auparavant, l'heuristique permettant d'effectuer le choix de la méthode de rendu à employer et, le cas échéant, le niveau de précision avec lequel doit être effectué le rendu, a été développée en s'appuyant sur une étude de la qualité du lancer de rayons, suivant différentes conditions d'utilisations.

3.1 Protocole de tests

Comme le lancer de rayons employé ne garantit pas une intersection exacte entre le rayon et le champ de hauteurs, il peut introduire un certain nombre d'artéfacts de rendu, tels que des effets de crénelage, comme l'illustre la figure 7.3. L'heuristique qui permet de moduler le niveau de précision du rendu de notre méthode se doit de prendre en compte ces artéfacts afin d'optimiser le rendu. Nous avons donc souhaité, dans un premier temps, évaluer ces différents problèmes. Pour cela, nous avons mis en place un protocole de tests, destiné à calculer une mesure d'erreurs sur la géométrie de la surface, générée par le lancer de rayons, par rapport au rendu d'un maillage de référence.

Paramètres de visualisation L'algorithme évalué dans cette étude est l'algorithme employé pour effectuer le rendu par lancer de rayons dans notre méthode hybride. Sa qualité visuelle a été mesurée sur différents jeux de données pour des paramètres de visualisation particuliers où varient l'angle de vue (de rasant à orthogonale au plan de base du champ de hauteurs) et la distance

1. Il faut toutefois noter que certaines contraintes matérielles peuvent entrer ici en jeu, telles que les accès mémoires, si les données sont de trop grande taille.



(a) Rendu de référence (b) Lancer de rayons

Figure 7.3 – Artéfacts de rendu introduits par un lancer de rayons dont les étapes de recherche d'intersections ne sont pas suffisamment précises.

du point de vue. Pour chaque position, une mesure d'erreur est effectuée pour différents nombres d'itérations, définissant la précision de la recherche d'intersection le long des rayons lancés. Nous nous sommes limités à deux niveaux de précision extrême dont les valeurs paramétrant la précision, ont été fixées arbitrairement pour obtenir un rendu dégradé et un rendu précis de la surface des jeux de données de tests :

Faible précision utilise 5 pas de recherche linéaire et 2 pas de recherche dichotomique ;

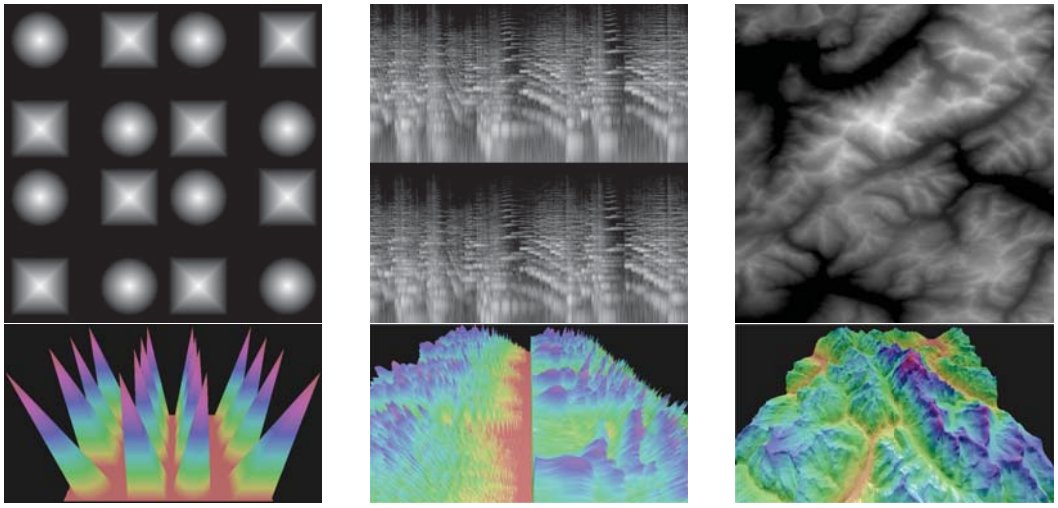
Haute précision utilise 64 pas de recherche linéaire et 25 pas de recherche dichotomique.

Données utilisées Les champs de hauteurs utilisés, ainsi qu'une vue de leur rendu, sont illustrés par la figure 7.4. Chacun de ces jeux de données, utilisés pour effectuer ces tests, présentent des caractéristiques très précises.

Hautes amplitudes Le relief de ce jeu de données synthétiques à la particularité de présenter de très fortes amplitudes. Il est composé de 2048×2048 élévations.

Spectrogramme Ces données représentent un spectrogramme et se rapprochent de données scientifiques. Ces données présentent une fréquence élevée de petits reliefs, c'est-à-dire que la surface du champ de hauteurs est composée d'une multitude de petits pics. Sa taille est de 4096×4096 élévations.

Mont Blanc Ce jeu de données topographiques représentent la région des Alpes où se situe le massif du Mont Blanc. Ce jeu de données re-



(a) Hautes amplitudes (b) Spectrogramme (c) Mont Blanc

Figure 7.4 – Champs de hauteurs, et leur rendu correspondant, utilisés pour évaluer la qualité du lancer de rayons.

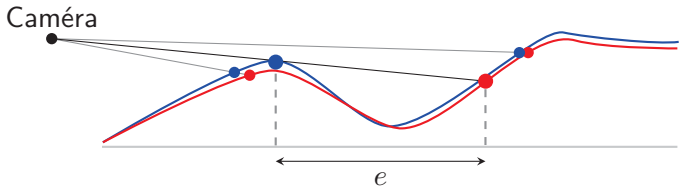


Figure 7.5 – Mesure de distance pour le calcul de l'erreur. En rouge la surface de référence, en bleu la surface générée par le lancer de rayons.

présente un terrain standard. Il est également composé de 4096×4096 élévations.

Ces différents jeux de données permettent ainsi de couvrir un panel d'utilisations assez vaste pour le lancer de rayons GPU, et plus spécialement dans des cas extrêmes d'utilisations : jeu de données à hautes amplitudes pour mettre en évidence la préservation des silhouettes et spectrogramme pour mettre en exergue la conservation des détails fins et à hautes fréquences d'un relief.

Métrique d'erreur Comme métrique d'erreur pour mesurer la qualité du rendu, nous proposons d'utiliser la distance, mesurée le long du rayon de vue, entre la surface issue d'un rendu de référence des données (via un maillage construit en associant chaque élévation de la carte de hauteurs à un sommet) et la surface obtenue via le lancer de rayons à évaluer. Cette distance est illustrée par la figure 7.5. Une erreur de ce type est assez bien adaptée pour évaluer un lancer de rayons, car elle permet de fournir une interprétation assez intuitive des mesures, pour chaque pixel qui compose l'image. En effet, une

valeur d'erreur nulle signifie que, comparée au maillage de référence, aucune erreur sur le rendu n'est commise tandis qu'une valeur qui tend vers l'infinie signifie que certaines parties du rendu sont totalement manquantes.

Pour évaluer de façon générale la précision du rendu sur une image, une valeur de *PSNR* (*peak signal-to-noise ratio*) est calculée pour chaque image, en suivant l'équation suivante :

$$E = 10 \cdot \log_{10} \left(\frac{R^2}{MSE} \right)$$

où l'erreur quadratique moyenne MSE est dérivée de l'erreur e_i de tous les pixels composant l'image F évaluée par l'équation :

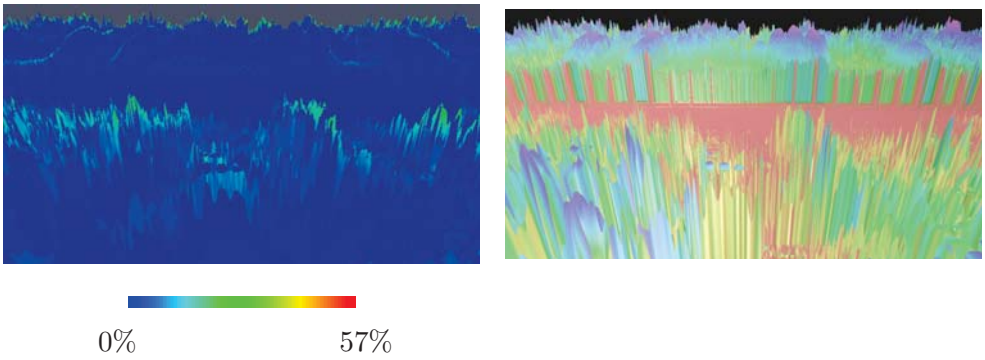
$$MSE = \frac{1}{\#F} \sum_{i \in F} e_i^2$$

Cette erreur dépend du nombre de pixels $\#F$ formant l'image ainsi que de la longueur R. Elle est définie par la plus longue diagonale du volume englobant l'ensemble des données. L'erreur e_i représente donc simplement la distance, sur le rayon de vue entre la surface de référence du champ de hauteurs et celle rendue via le lancer de rayons (*cf.* figure 7.5).

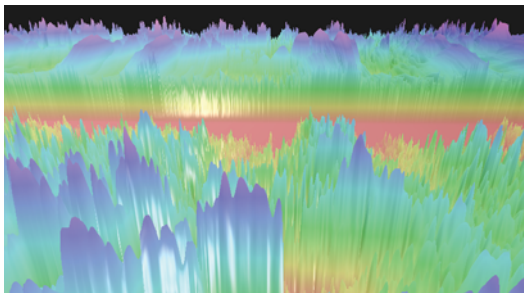
3.2 Résultats

La figure 7.6 montre un exemple de distribution spatiale de l'erreur sur une image rendue avec un lancer de rayons de faible précision pour un angle de vue rasant. Comme cela est visible sur cette figure, comme certaines intersections sont manquées au niveaux des lignes de crêtes du relief, il en résulte des valeurs d'erreurs élevées. Par conséquent, cela implique une dégradation assez forte des silhouettes du relief. Ces artéfacts sont fortement réduits pour devenir négligeables lorsque la précision du lancer de rayons augmente suffisamment. On peut également noter que sur cette figure 7.6, la valeur de PSNR est de 51,87 dB. Elle est distribuée de la façon suivante : $69,5\%$ des pixels présentent une erreur de moins de 1% de la longueur du terrain, 80% des pixels ont une erreur de moins de 5% de cette longueur et enfin $89,5\%$ des pixels de moins de 10%.

En ce qui concerne la figure 7.7, on peut observer que l'influence majeure sur la qualité du lancer de rayons est l'angle de vue. Dans le cas où le terrain présente un relief assez lisse, l'erreur présente une dépendance linéaire avec cet angle de vue. Il est aussi intéressant de noter que la distance ne semble pas introduire d'artéfacts de rendu. En effet, si on se réfère toujours à cette figure



(a) Lancer de rayons faible précision

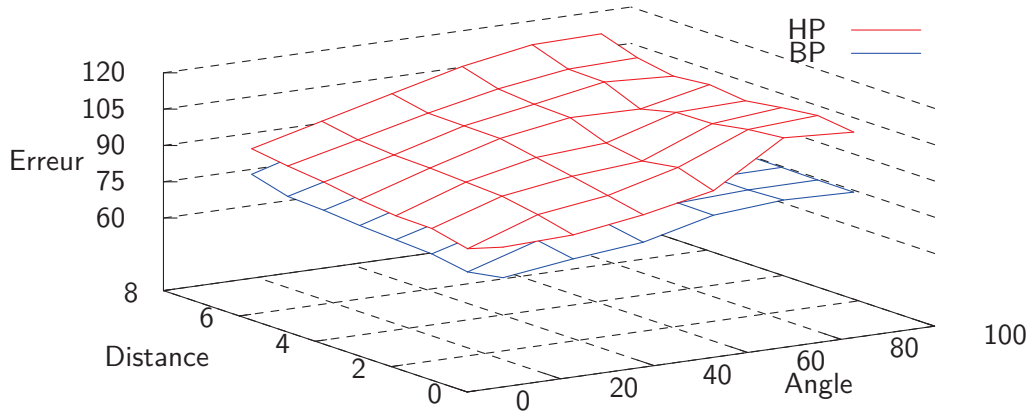


(b) Rendu de référence

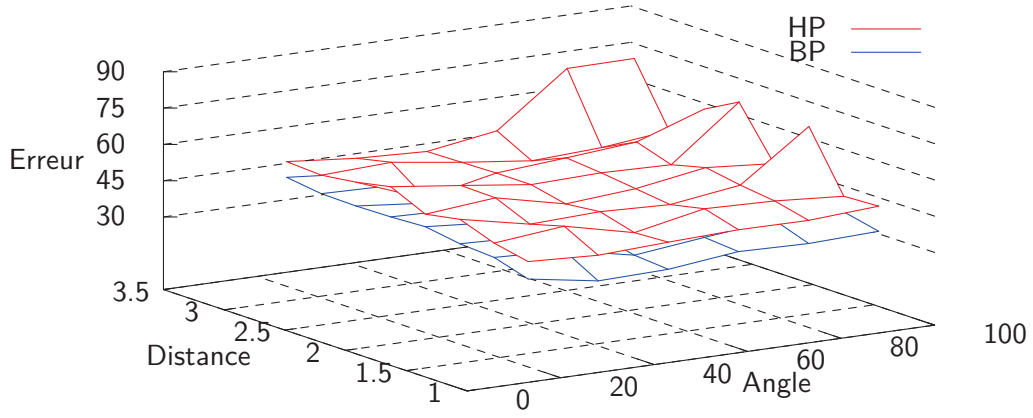
Figure 7.6 – Erreur géométrique du rendu par un lancer de rayons de faible précision (a) comparé à un maillage de référence (b) pour le jeu de données "spectrogramme". L'erreur donnée est relative à la taille du terrain.

7.7, la qualité de rendu est plutôt constante en fonction de la distance. Par contre, l'influence du relief joue un rôle important quand celui-ci présente de fortes amplitudes. Cela est particulièrement vrai sur notre jeu de données à fortes amplitudes (*cf.* figure 7.7b), où les erreurs sont toujours présentes, même lorsque les données sont observées avec un angle de vue presque orthogonal par rapport au plan de base de la carte d'élévations.

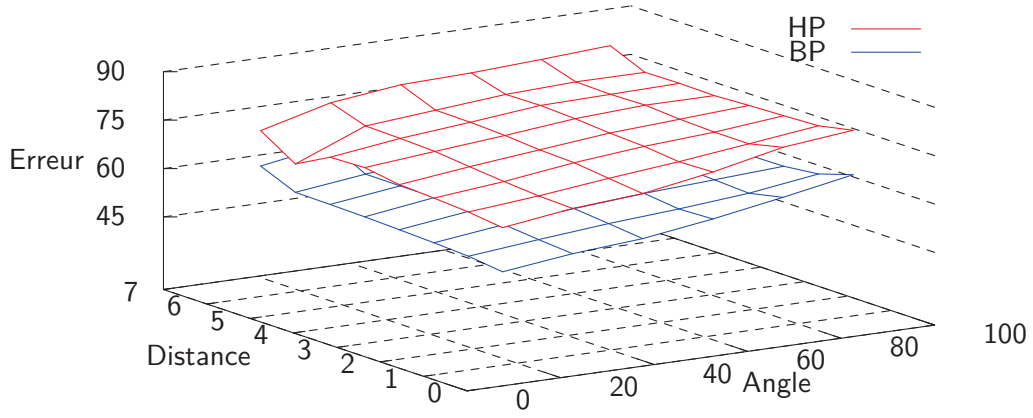
On peut encore noter un point intéressant de ces différentes expérimentations. En effet, on peut voir que la taille des données n'a pas un impact direct sur les performances de rendu, et en particulier en ce qui concerne la vitesse de rendu. Cette constatation est mise en évidence par les vitesses mesurées et présentées dans le tableau 7.1. Cela s'explique par le fait que l'algorithme de lancer de rayons dépend principalement du nombre de pixels qui sont effectivement rendus. Par contre, la vitesse est étroitement liée à la précision des étapes de recherche d'intersections du lancer de rayons. Cependant, et comme l'illustre la figure 7.6a, si la précision du lancer de rayons n'est pas suffisante, les artéfacts de rendu deviennent très visibles, notamment au premier plan. Il faut également ajouter que ces artéfacts sont encore plus sensibles lorsque la



(a) Mont Blanc



(b) Hautes amplitudes



(c) Spectrogramme

Figure 7.7 – Mesures de PSNR pour deux niveaux de précisions du lancer de rayons (haute précision, HP, en rouge et basse précision, BP, en bleu) sur les trois jeux de données tests. Ces mesures sont effectuées en faisant varier la distance du point de vue et l'angle de vue.

Jeux de données (Taille des données)	Hautes Amplitudes (2048^2)	Mont Blanc (4096^2)	Spectro-gramme (4096^2)
Lancer de rayons basse précision	430	360	238
Lancer de rayons haute précision	76	47	36
Rendu maillage	26	6,9	7

Tableau 7.1 – Vitesse de rendu (en Hz) pour un angle de vue rasant, sur une fenêtre de 1024×768.

position du point de vue est déplacée. On peut en conclure que le lancer de rayons à faible précision doit être utilisé précautionneusement et que l'efficacité de la méthode passe nécessairement par un bon équilibre entre la qualité de rendu et les performances.

4 Heuristique du choix du niveau de précision

Les observations faites dans la section précédente sur les propriétés du lancer de rayons, suivant les différents usages, sont à la base de la définition de l'heuristique qui permet au mécanisme adaptatif, de notre méthode hybride, de faire le choix de la technique de rendu à employer pour une cellule et, le cas échéant, du niveau de précision à utiliser. Ce choix doit être fait pour maintenir un haut niveau de qualité avec une vitesse de rendu élevée.

4.1 Choix de la technique de rendu

Comme nous l'avons montré dans la section précédente, l'algorithme de lancer de rayons est capable soit d'offrir une qualité de rendu très élevée, soit d'atteindre une vitesse de rendu élevée. Cela dépend essentiellement de la précision des recherches d'intersections entre rayons et surface du champ de hauteurs. De plus, utiliser un maillage dont le nombre de sommets correspond au nombre d'élévations de la carte de hauteurs, permet d'obtenir le rendu le plus précis possible. Pour une distance de vue proche, le rendu par maillage est également la méthode la plus performante, à partir du moment où un triangle se projette sur une surface assez importante de l'écran. Il faut toutefois noter que, dans ce cas, la vitesse d'affichage est fortement dépendante des capacités du matériel graphique. En tenant compte de ce point, nous proposons d'introduire un seuil basé sur la couverture du terrain à l'écran pour choisir entre le rendu par rastérisation d'un maillage et le lancer de rayons. Ainsi, un paramètre, nommé T, est défini pour tenir compte des capacités matérielles

du processeur graphique. Son impact doit être le suivant sur le mécanisme adaptatif : lorsque le rendu par maillage devient plus efficace que le lancer de rayons, alors l'heuristique doit choisir la rastérisation du terrain. Intuitivement, ce seuil doit s'apparenter à la surface des triangles projetés à l'écran, ainsi, quand la surface de ceux-ci est suffisamment grande, le rendu par maillage sera plus efficace.

D'un point de vue technique, l'implantation suit l'idée suivante : pour chaque cellule, la boîte qui lui est associée, est projetée à l'écran. Ainsi, la surface qu'elle couvre peut être calculée. Afin de simplifier le calcul de cette surface, nous utilisons simplement une approximation de cette dernière en calculant, en espace image, l'aire $\mathcal{A}$ du rectangle englobant les huit sommets projetés de la boîte. Cette valeur permet de combiner la distance, entre la caméra et la cellule, avec la variation du relief au sein de la cellule. Cette variation est représentée par les valeurs de l'élévation minimale et de l'élévation maximale de la cellule. En fait, comme nous sommes intéressés par la taille des triangles à l'écran, l'aire $\mathcal{A}$ est divisée par le nombre d'élévations S contenues par la cellule. En d'autres termes, S est défini par la taille des cellules (s'il y a n échantillons sur le côté d'une cellule, $S = n^2$). Le choix de la méthode de rendu à employer se fait donc en évaluant le ratio $\mathcal{A}/S$. S'il est plus grand que le paramètre T, alors le maillage doit être utilisé pour la cellule, sinon, on utilise le lancer de rayons et on calcule son niveau de précision. Ce choix se résume donc ainsi :

$$\begin{cases} \frac{\mathcal{A}}{S} > T \Rightarrow \text{rastérisation} \\ \frac{\mathcal{A}}{S} \leq T \Rightarrow \text{lancer de rayons} \end{cases}$$

Dans le cas où le lancer de rayons est sélectionné, on calcule son niveau de précision.

4.2 Calcul du niveau de précision du lancer de rayons

Afin d'optimiser la vitesse d'affichage, nous souhaitons utiliser le minimum de pas lors de la recherche d'intersections, effectué par le lancer de rayons. Cependant, comme cela est décrit à la section 3, cet algorithme introduit des artéfacts de rendu assez importants lorsque l'angle de vue est rasant. Ces derniers doivent être limités au maximum. Comme nous l'avons également montré précédemment, l'amplitude des reliefs à l'intérieur d'une cellule est aussi un paramètre important dont il faut tenir compte. En effet, si la cellule représente une part importante de la vue (*i.e.* la boîte qui lui est associée couvre une surface importante de l'écran), la précision du lancer de rayons doit être accrue. De façon similaire, la précision peut être diminuée en fonction de la distance de vue sans trop impacter la qualité finale du rendu.

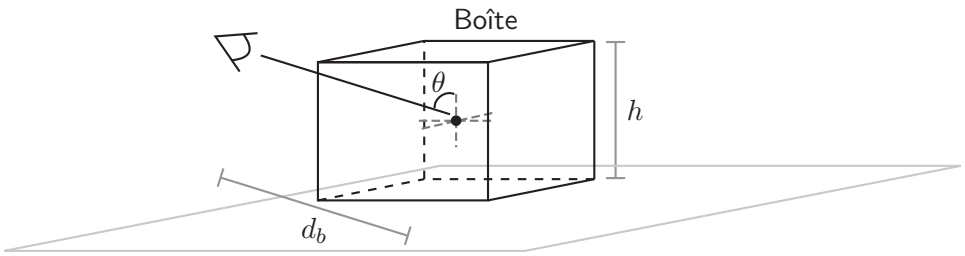


Figure 7.8 – Schéma représentant les paramètres utilisés pour le calcul de l'heuristique.

Par conséquent, l'angle de vue, la distance entre la cellule et la caméra ainsi que l'amplitude du relief à l'intérieur de la cellule sont inclus dans notre heuristique. Ces différents paramètres sont illustrés par la figure 7.8. Pour calculer le niveau de précision d'une cellule donnée, nous proposons d'utiliser la formule suivante :

$$L = \frac{min(d_b, d_{max})}{d_{max}} - (h \cdot \cos\theta)$$

Dans cette équation, h représente la hauteur de la boîte[2], θ est l'angle entre le rayon de vue et la verticale, au centre de la boîte tandis que d_b représente la distance entre la caméra et le centre de la boîte. La valeur d_{max} permet de borner les valeurs prises par L. Elle est calculée comme suit :

$$d_{max} = k \cdot d_{BB}$$

La valeur d_{BB} représente la valeur maximale de d_b qui correspond à la longueur de la plus grande diagonale de la boîte englobante du terrain, exprimée en espace objet. Le coefficient k est introduit pour moduler la distribution des niveaux de précisions en fonction de la distance entre la caméra et la cellule.

Suivant cette définition de l'heuristique, on constate qu'elle peut se décomposer en deux termes. Le premier permet de contrôler la variation des niveaux de précision liés à la distance d'observation. Le second terme, quant à lui, permet cette même variation mais en fonction de l'aspect de la boîte englobant le relief de la cellule. En d'autres termes, il agit sur la précision en fonction de l'angle de vue par rapport au plan de base du terrain et en fonction de l'amplitude du relief dans la cellule concernée.

Afin d'exploiter les valeurs calculées pour définir le niveau de précision à employer, les valeurs de L sont uniformément distribuées entre les différents niveaux de précisions. La valeur 0 représente ainsi le niveau de précision le plus fin.

2. On suppose que les élévations de la carte de hauteurs sont comprises dans l'intervalle $[0; 1]$.

5 Notes sur l'implantation de la méthode hybride

Afin de tester la méthode de rendu proposée, nous l'avons implantée. Cette implantation a recours essentiellement au processeur graphique, comme le suggèrent les diagrammes constituant les figures 7.1 page 117 et 7.9. Le diagramme de la figure 7.9 donne un aperçu plus précis de l'implantation des différentes étapes de la méthode de visualisation ainsi que les différents types de stockage des données utilisés. La phase de calculs permettant de déterminer l'algorithme de rendu à employer pour une cellule, de même que le calcul du niveau de précision du lancer de rayons, sont ainsi implantés au sein d'un *vertex shader*, en suivant l'algorithme 7.1.

Chaque algorithme de rendu est exécuté également sur le processeur graphique (principalement *vertex shader* pour le placage de déplacement et essentiellement le *fragment shader* pour le lancer de rayons). On peut cependant noter que l'étape de génération de la géométrie des boîtes, servant de base au lancer de rayons, est effectuée par *geometry shader*. La seule étape qui demeure sur le processeur central, est le tri des cellules, mené à partir du résultat du calcul du niveau de précisions. Cette étape ne peut pas être implantée efficacement sur le matériel graphique dont nous disposions au moment où l'implantation a été effectuée. Cependant, il semble qu'avec les toutes dernières générations de processeurs graphiques [3], cette restriction soit levée, ce qui laisse présager un possible accroissement des performances globales de la méthode de rendu.

Concernant le stockage des données, pour chaque cellule, la carte de hauteurs, qui peut également contenir en plus les vecteurs normaux de la surface en chaque élévation, est nécessaire. De plus, les élévations maximales et minimales de chaque cellule sont requises. On peut noter que les vecteurs normaux ne sont pas obligatoirement stockés et peuvent simplement être calculés à la volée, au moment du calcul d'illumination. Une information permettant de définir la division cellulaire doit également être mémorisée. L'implantation, sur le *geometry shader*, de la génération des boîtes, englobant le relief de chaque cellule et nécessaires au lancer de rayons, permet de limiter les données utiles à la modélisation de la division cellulaire à seulement un tableaux de sommets. En effet, il n'est nécessaire que de stocker la position de l'origine des cellules, définie par des coordonnées à deux dimensions, exprimées sur le plan de base du champ de hauteurs.

Les cartes d'élévations et de normales ainsi que les altitudes minimales et maximales de chaque cellule, sont stockées dans deux textures dont la dis-

3. Processeurs graphiques basés sur une architecture de type NVIDIA Fermi, par exemple.

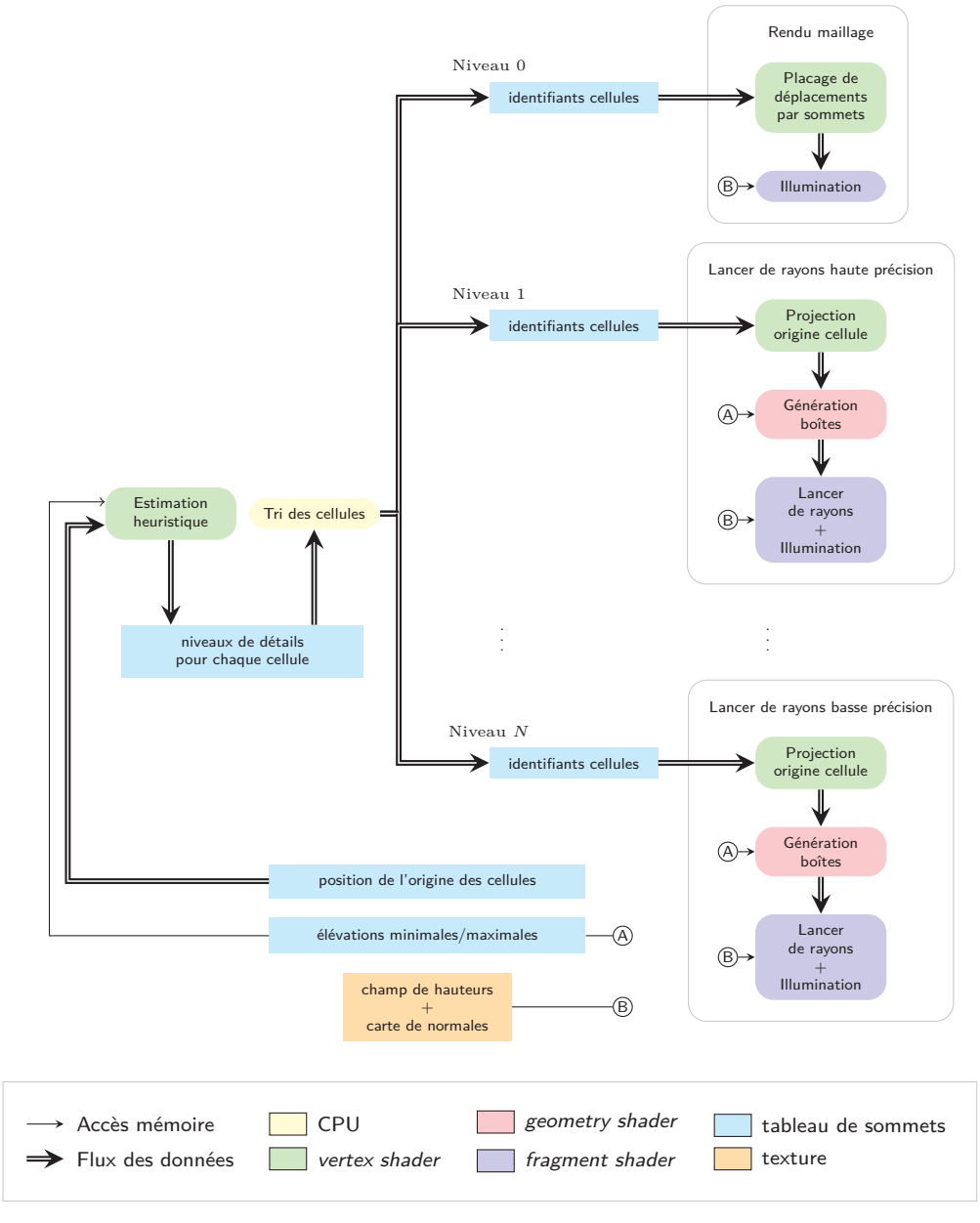


Figure 7.9 – Flux des données et types de stockage utilisés dans la chaîne de rendu.

Entrées :
`// Variables associées à chaque cellule (attributs de sommets)`
centreCellule ← position du centre de la boîte ;
boîte ← sommets formant la boîte associée à la cellule ;
h_{min} ← élévation minimale contenue dans la cellule ;
h_{max} ← élévation maximale contenue dans la cellule ;
`// Variables valables pour toutes les cellules (variables uniformes ou constantes)`
T ← paramètre utilisateur (module la couverture du maillage);
k ← paramètre utilisateur (module la distribution des niveaux de précision du lancer de rayons);
nbNdP ← nombre de niveaux de précisions ;
S ← nombre d'échantillons par cellule ;
diagonaleBoiteEnblobante ← longueur de la plus longue diagonale de la boîte englobant toutes les données;
`// Variables dépendant de l'image courante mais commune à chaque cellule (variables uniformes)`
caméra ← position de la caméra ;

Sorties :
niveauDePrécision : niveau de précisions pour la cellule courante (variable *varying*);

A ← `calculAireBoiteProjetée` (boîte) ;
θ ← `calculAngleParRapportVerticale`(caméra, centreCellule);
ratioTailleTriangle ← A/S

si ratioTailleTriangle $\geqslant$ T **alors**
  niveauDePrécision ← 0;
sinon
  distanceCaméra ← `distance`(caméra, centreCellule);
  distanceCaméra
  ← `min`(distanceCaméra, k × diagonaleBoiteEnblobante);
  distanceCaméra ← distanceCaméra/(k × diagonaleBoiteEnblobante);
  niveauDePrécision ← distanceCaméra $-$ $(\mathsf{h}_{max} - \mathsf{h}_{min}) \times \cos(\theta)$;
  niveauDePrécision ← $\frac{\text{niveauDePrécision}}{\text{k}-1} \times (\text{nbNdP} + 1)$;
  niveauDePrécision ← `entierLePlusProche` (niveauDePrécision) ;
fin

retourner niveauDePrécision ;

Algorithme 7.1 – Mécanisme du choix du niveau de précision du rendu, implanté sur GPU au niveau d'un *vertex shader*.

position correspond à celle du terrain. Ce schéma de stockage mémoire a été choisi car il est simple à mettre en place. Néanmoins, il est assez limitatif car il ne permet pas de prendre en charge des jeux de données de trop grande taille : il faut que toutes les données puissent résider en mémoire graphique. Cependant, comme nous utilisons une division cellulaire des données pour y accéder et les rendre, il est tout-à-fait possible d'ajouter un mécanisme de chargement dynamique des données, en fonction des besoins.

Comme nous l'avons déjà abordé au cours de la section 2.2, et dans le but de pouvoir visualiser des données dynamiques, aucune étape de prétraitements des données n'est effectuée. Par conséquent, l'algorithme de lancer de rayons que nous employons n'inclut pas de mécanisme d'optimisation, tels que les mécanismes décrits à la section 7 page 43. Toutefois, il est parfaitement envisageable d'incorporer, à notre méthode, de tels mécanismes, au prix d'une consommation mémoire accrue et de l'impossibilité de visualiser des jeux de données dynamiques.

Données dynamiques : mécanismes de mise-à-jour des données Pour illustrer le fonctionnement de cette méthode avec des données dynamiques, différents mécanismes simples ont été implantés. Parmi ces mécanismes, nous avons une simulation basique d'érosion de terrains, un mécanisme d'édition des données et une simulation basique de vagues d'océans.

La simulation d'érosion est entièrement implantée sur le processeur graphique. Elle est issue de travaux de Olsen [Ols04]. Cette simulation s'applique à l'ensemble des données représentant le terrain et les modifie à chaque pas de simulation. Les données recalculées sont les élévations de la carte de hauteurs, les altitudes maximales et minimales de chaque cellule et les vecteurs normaux. De façon similaire, la simulation de vagues d'océans suit le même principe en modifiant à chaque pas les mêmes données et s'applique également à l'échelle du terrain. Son principe est extrêmement basique et ne se limite qu'à une simple composition de fonctions sinusoïdales, appliquées sur les élévations, en fonction de leur position sur la carte de hauteurs.

Pour démontrer les possibilités de modifications locales des données, le mécanisme d'édition permet de déformer localement le terrain, à partir d'un motif représenté par un champ de hauteurs. Lors de ces modifications, seule la zone concernée de ce dernier (incluant également la carte de normales) est mise à jour. Il en est de même pour les élévations maximales et minimales des cellules. Seules les cellules impactées par l'édition sont modifiées. Ce processus est implanté sur le processeur central mais dans un processus séparé, ce qui permet de conserver une visualisation active et ainsi d'animer la modification des données, tout en gardant la visualisation temps réel.

Chapitre 8

Analyse de l'approche hybride pour le rendu de terrains

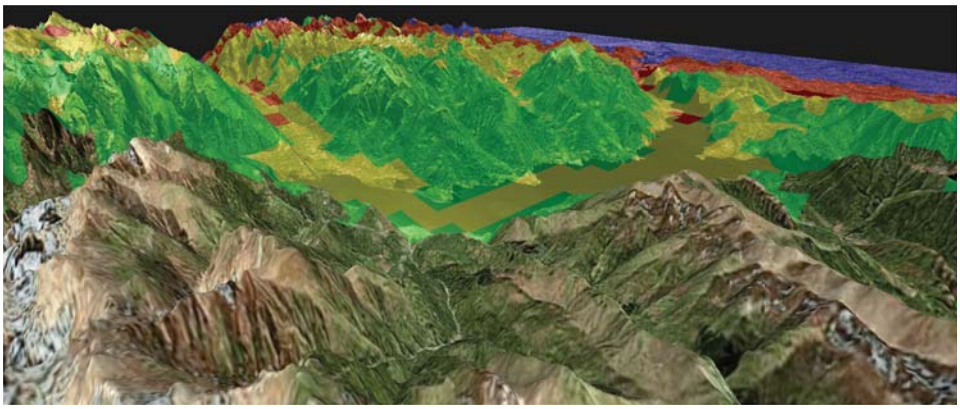
Dans ce chapitre, nous décrivons et analysons en détails les performances de la méthode hybride de rendu de terrains que nous avons décrite au chapitre précédent. Ces analyses s'appuient sur différentes mesures de performances, aussi bien en termes de qualité que de vitesse. Les mesures de performances données, ont été mesurées sur un ordinateur de type PC standard, doté d'une carte graphique NVIDIA GeForce GTX 280 embarquant 1 Go de mémoire graphique.

Dans un premier temps, nous présentons quelques résultats qualitatifs et de vitesse à la section 1. Afin de compléter ces résultats, nous proposons, au cours de la section 2, de comparer notre méthode avec les *geometry clipmaps* [LH04]. La prise en compte des données dynamiques étant un atout majeur de notre méthode, des résultats sur ce type de données sont également présentés à la section 3, avant de dresser un premier bilan de cette méthode pour conclure ce chapitre à la section 4.

1 Performances et qualité du rendu

La première série de résultats présentés s'attache à montrer principalement les résultats visuels et les performances de rendu obtenus. Ces résultats sont bien évidemment assez dépendants des paramètres de l'heuristique. Leur impact sur les performances générales de la méthode a également été étudié.

(a) Vue générale



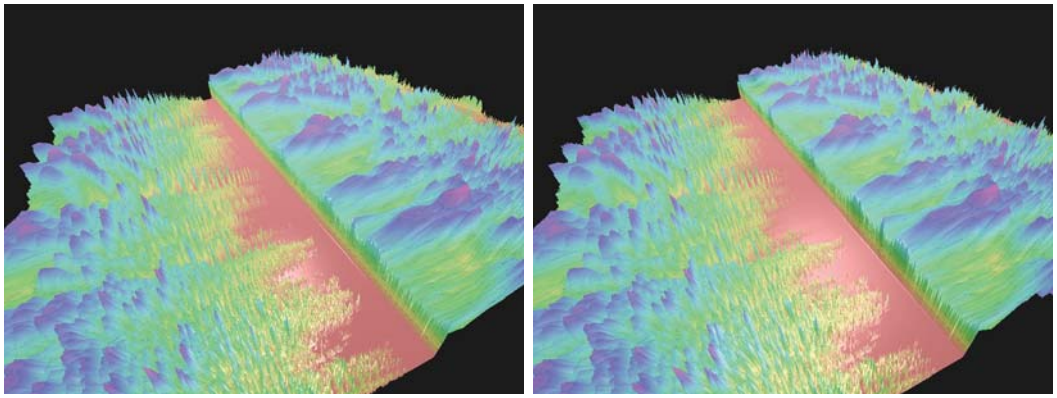
(b) Distribution des niveaux de précisions

Figure 8.1 – Vue globale d'un rendu de la région du lac de Côme dans les Alpes Italiennes, associée à sa texture de vues satellites de la région.

1.1 Résultats qualitatifs

La figure 8.1 présente une capture d'un rendu que permet d'obtenir la méthode développée, sur un jeu de données topographiques associé à une texture couleurs des vues satellites de la région en question. Ce jeu de données a une résolution de 4096×4096 pixels, de même que la texture couleurs associée. Le rendu de cette image est effectué sur une fenêtre de taille 1900×700 pixels à une vitesse d'environ 43 Hz. Il utilise quatre niveaux de précision pour le lancer de rayons, dont les nombres de pas effectués pour les étapes de recherche linéaire et dichotomique sont les suivants :

1. 55 linéaires et 25 binaires ;



(a) Maillage de référence **(b)** Notre méthode

Figure 8.2 – Comparaison entre un rendu par maillage et notre méthode.

2. 32 linéaires et 15 binaires ;
3. 16 linéaires et 10 binaires ;
4. 8 linéaires et 5 binaires.

Les surimpressions colorées de la figure 8.1b matérialisent la distribution des niveaux de précision sur ce rendu. Cela montre assez clairement que la méthode développée, guidée par l'heuristique, s'adapte au relief des données pour moduler la précision du rendu, et plus particulièrement celle du lancer de rayons. Pour effectuer ce rendu, nous avons utilisé les paramètres suivant pour l'heuristique : $T = 70$ et $k = 1,30$. La division cellulaire des données est faite sur la base d'une taille de cellules de 128×128 échantillons.

En utilisant la configuration de rendu décrite ci-dessus, un survol de ces données s'effectuent avec un taux de rafraichissement compris entre 40 et 150 Hz. Ces valeurs dépendent évidemment de la couverture du terrain à l'écran, et par conséquent des différentes positions de la caméra. À titre de comparaison, en rendant ces données avec un maillage à pleine résolution, mais sans mécanisme de niveaux de détails, nous obtenons une vitesse de rendu comprise entre 5 et 9 Hz.

En ce qui concerne la qualité de rendu, un aperçu de celle-ci est donné par les figures 8.2 et 8.9 page 147, qui comparent un rendu via notre méthode et l'image correspondante, en utilisant un maillage à pleine résolution, sur le jeu de données représentant un spectrogramme. On constate sur ce rendu que l'ensemble des petits reliefs composant ces données est bien conservé par notre méthode.

Un des plus gros problèmes avec les méthodes combinant différentes techniques de rendu, est le changement brusque d'algorithmes de rendu, ou de niveaux de détails, ou encore de précision du rendu. Cela se traduit visuellement par des artéfacts (*popping*) lors du changement brusque du rendu de différentes parties des objets visualisés. Cependant, avec notre méthode de rendu, à partir du moment où on utilise une précision suffisante pour les différents lancers de rayons, ce type de problèmes n'est pas visible.

De façon similaire, notre méthode n'introduit pas de fissures entre les différentes cellules rendues. Habituellement, l'utilisation d'une division cellulaire et le rendu indépendant de chacune d'elles est favorable à l'apparition de ce type d'artéfacts. Néanmoins, toutes les cellules rendues via un maillage le sont avec un maillage de résolution identique, aucune jonction en T n'est introduite aux bordures entre deux cellules maillées. De même, dans le cas des frontières entre cellules maillées et cellules rendues par lancer de rayons, aucune fissure n'est visible. La raison en est la suivante : le lancer de rayons opère, comme pour le rendu par maillage, sur les données à pleine résolution, par conséquent, le point d'entrée du rayon, dans la boîte servant de support au rendu, est toujours cohérent par rapport au maillage des cellules voisines. De plus, le même taux d'échantillonnage ainsi que la même interpolation entre les échantillons sont utilisés pour chaque cellule. Ceci représente un avantage non négligeable de notre méthode par rapport aux méthodes de rendu basées sur des maillages multi-résolutions qui se voient fréquemment confrontées à ces fissures, aux limites entre différentes parties du terrain rendues à différentes résolutions. Dans ces cas, les solutions pour pallier à ces problèmes sont complexes et nécessitent de recourir à des dispositifs de remaillage ou de subdivisions adaptatives.

Toujours sur le plan de la qualité visuelle du rendu, comme nous n'employons aucun mécanisme de niveaux de détails géométrique, la méthode développée ne propose aucun mécanisme pour résoudre les problèmes liés à l'aliasing. Cependant, on peut noter que les résultats expérimentaux n'ont pas révélé ce type de problèmes de manière trop prononcée. Ils sont essentiellement localisés sur les zones où le relief est rendu avec les plus faibles niveaux de précision. Dans les faits, les cellules rendues avec les plus hauts niveaux de précision et par le maillage ne souffrent pas de ces problèmes d'aliasing. En effet, le maillage est principalement utilisé sur les zones du terrain qui sont proches du point de vue. Par conséquent, la taille des triangles affichés à l'écran est suffisamment grande pour éviter ces artéfacts. Par contre, pour les zones plus lointaines, un faible aliasing reste visible, mais ne peut-être évité sans un mécanisme dédié qui se révèle généralement coûteux et complexe à mettre en place.

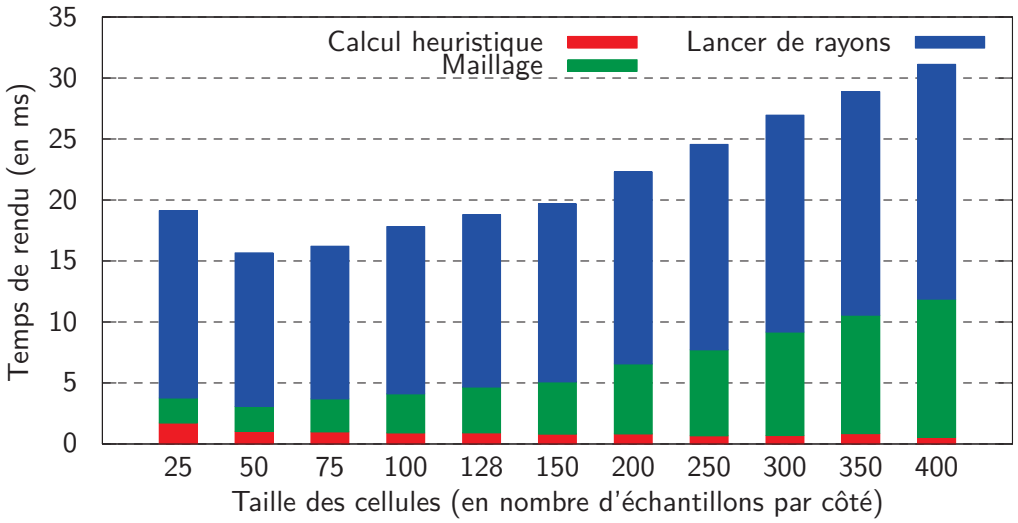


Figure 8.3 – Influence de la taille des cellules sur les temps de rendu des différentes étapes de notre méthode. Ces temps sont mesurés sur le jeu de données représentant le massif du Mont-Blanc.

1.2 Influence des paramètres de l'heuristique

Les résultats et mesures présentés dans la section précédente sont fortement liés aux différents paramètres qui modifient le comportement de l'heuristique de choix du niveau de précision. Il convient d'en analyser l'impact sur les performances générales de rendu. Parmi ces derniers, on retrouve la taille des cellules, le nombre de pas de recherche pour effectuer le lancer de rayons, le nombre de niveaux de précision et les deux paramètres utilisateurs T et k offrant la possibilité de moduler l'heuristique.

Taille des cellules La taille choisie pour effectuer la division cellulaire des données, impacte directement sur les performances de rendu. La figure 8.3 donne un aperçu du temps que prennent les différentes phases de notre méthode de rendu, pour différentes tailles de cellules, variant de 50×50 échantillons à 128×128 échantillons par cellules. On peut constater que lorsque la taille des boîtes devient trop petite, les performances de rendu décroissent. Dans ce cas, trop de cellules doivent être rendues via le lancer de rayons ce qui implique un recalcul fréquent des mêmes pixels aux voisinages de celles-ci. À l'opposé, lorsque la taille des boîtes augmente, la surface du terrain rendu par un maillage devient plus importante et est donc plus lente à rendre. De plus les zones communes aux cellules voisines sont plus grandes, ce qui implique, là aussi, un fort taux de pixels devant être réévalués plusieurs fois de suite.

Nombre de niveaux de précision	Évaluation de l'heuristique	Rendu maillage	Rendu lancer de rayons	Temps total de rendu
2	0,38	2,38	10,80	13,61
4	0,42	2,33	12,13	14,95
6	0,38	2,33	13,04	15,80
8	0,38	2,33	13,83	16,60
10	0,38	2,35	14,66	17,45

Tableau 8.1 – Influence du nombre des niveaux de précision sur les performances de rendu. Ces mesures de temps sont données en milliseconde.

Nombre de niveaux de précision du lancer de rayons Le tableau 8.1 montre l'influence du nombre de niveaux de précisions du lancer de rayons sur les performances de rendu. Pour effectuer ces différentes mesures, nous avons utilisé un jeu de données topographique représentant la région du Puget Sound, d'une taille de 4096×4096 échantillons. La taille des cellules est de 75×75 échantillons. La scène a été rendue en utilisant un nombre variable de pas pour la recherche linéaire, compris entre 64 et 6. Pour la recherche linéaire, les différents niveaux de détails ont fait appel à un nombre de pas compris entre 50 et 5. Comme nous pouvons le voir sur les mesures de la table 8.1, ces paramètres n'influencent évidemment que sur la passe de rendu par lancer de rayons. Par conséquent, le temps consacré au rendu du maillage et à l'évaluation de l'heuristique reste constant sur ces différentes configurations. En ce qui concerne la passe de rendu des cellules via le lancer de rayons, on constate que le temps de rendu croît avec le nombre de niveaux de précision. La raison de cette baisse de performances n'est pas facilement identifiable et ne peut pas être facilement imputée à une partie spécifique de la chaîne de rendu. Toutefois, ce comportement pourrait être la conséquence de l'accroissement du nombre de changements de contextes entre les différents programmes de *shaders*, associés aux différents niveaux de précision.

En ce qui concerne l'effet du nombre de niveaux de détails sur la qualité visuelle du rendu, on peut se référer aux indications fournies par la figure 8.4. Elle présente une mesure de PSNR effectuée de façon analogue à celle décrite à la section 3 page 121, en fonction de la vitesse, pour un ensemble d'images. On constate, sur cette figure, que la qualité visuelle n'est pas significativement accrue lors d'une augmentation du nombre des niveaux de précision. Cela est due au fait que sur ces mesures, la borne maximale et la borne minimale définissant le nombre de pas employés, restent les mêmes. Seul varie le nombre de niveaux de détails intermédiaires. À partir de ces résultats, on peut en déduire que quatre niveaux de détails sont suffisants pour donner un bon équilibre entre vitesse de rendu et qualité.

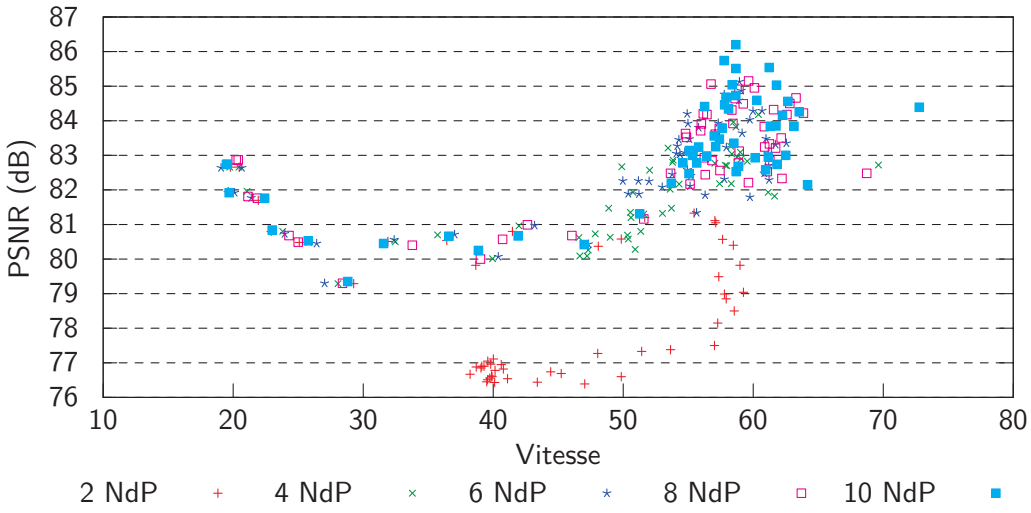
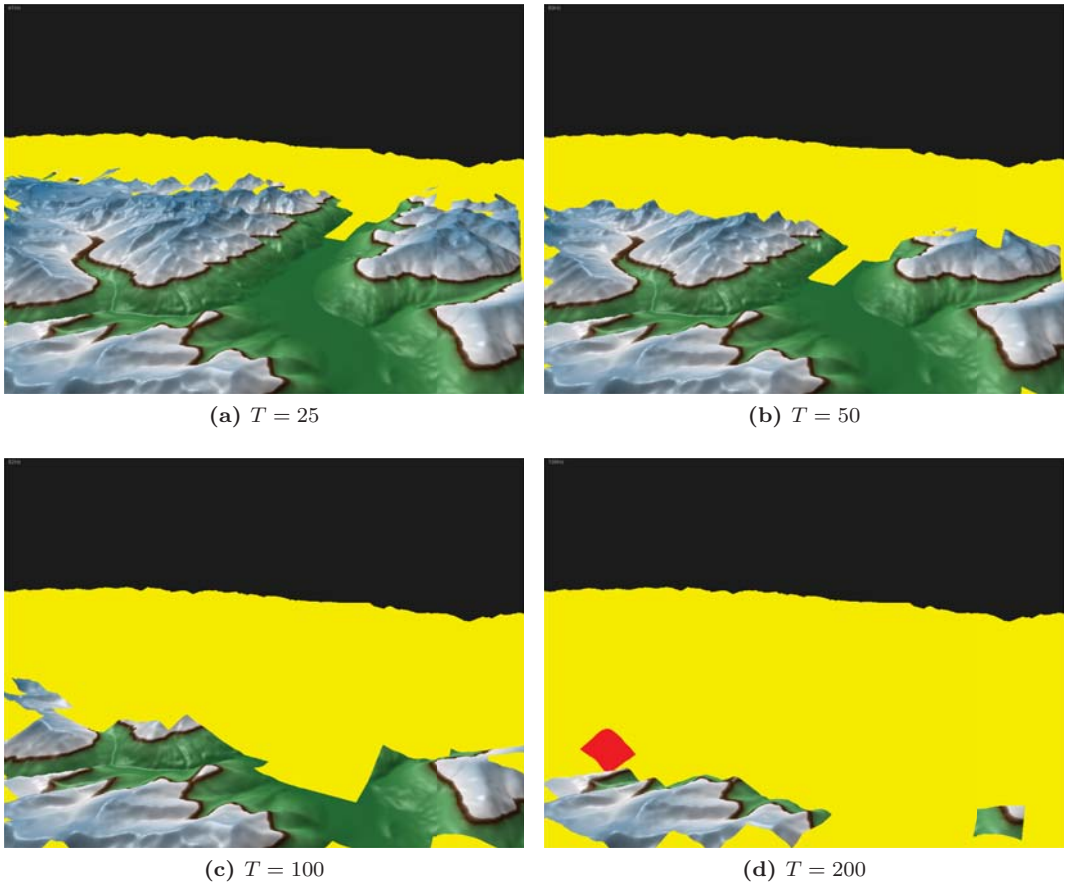


Figure 8.4 – Influence du nombre de niveaux de précision (NdP) sur la qualité du rendu. Cette figure donne, pour un ensemble d'images issue d'un survol du jeu de données représentant le Puget Sound, la valeur du PSNR en fonction de la vitesse de rendu.

Paramètres de l'heuristique En plus de l'influence des réglages généraux de la méthode de rendu décrite précédemment, l'heuristique admet deux paramètres qui permettent de modifier son comportement. Ces deux paramètres nommés T et k, dans sa description faite au chapitre 7, ont une influence sur le choix de la méthode de rendu à employer pour une cellule (paramètre T), ainsi que sur le choix du niveau de précision du lancer de rayons (paramètre k). Ces deux paramètres permettent ainsi d'adapter la méthode de visualisation aux données et également au matériel graphique disponible.

Le paramètre T est principalement utilisé pour définir si le maillage doit être employé ou non pour une cellule donnée. Il permet de sélectionner principalement les cellules du premier plan (ou présentant une large couverture de l'écran) pour les rendre en utilisant le maillage et ainsi utiliser une qualité optimale de rendu pour ces zones très visibles du terrain. Ainsi, quand le paramètre T croît, la couverture du maillage sur le terrain prend plus d'importance dans le choix de la méthode de rendu. Son comportement est illustré par la figure 8.5 qui présente une vue du jeu de données représentant la région des Dolomites, pour quatre valeurs différentes.

En ce qui concerne le paramètre k, son rôle permet d'adapter l'heuristique aux reliefs que décrit la carte d'élévations. Il agit en modifiant la distribution des niveaux de précision du lancer de rayons. En effet, lorsque le paramètre k prend des valeurs basses, un plus grand nombre de cellules sont rendues avec un niveau de précision bas. Par conséquent, le rendu est dégradé sur une plus



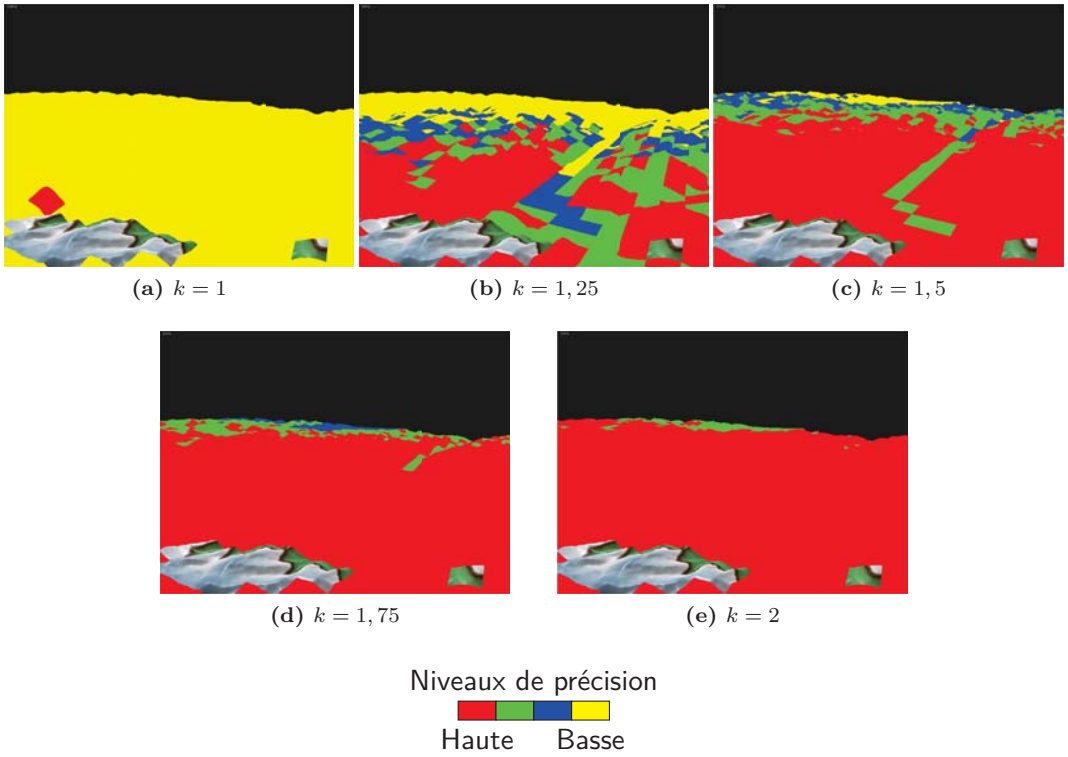
(a) $T = 25$ **(b)** $T = 50$

(c) $T = 100$ **(d)** $T = 200$

Figure 8.5 – Influence du paramètre T de l'heuristique. Les zones maillées sont les zones texturées des différentes captures. Plus le paramètre T prend de grandes valeurs, plus le nombre de cellules maillées est grand. Le paramètre k est fixe ($k = 1$).

grande partie du terrain. Le comportement de ce paramètre est illustré par la figure 8.6, en utilisant le même jeu de données que précédemment, en fixant plusieurs valeurs de k différentes.

En ce qui concerne l'influence de ces valeurs de paramètres sur la qualité et la vitesse du rendu, les figures 8.7 et 8.8 donnent un certain nombre d'indications. La qualité est exprimée avec une mesure de PSNR (figure 8.7) et avec une indication quant à la qualité du rendu (figure 8.8) et plus particulièrement celle des lignes de crêtes, en proposant une mesure du nombre de pixels manquées (*i.e.* présent sur le rendu de référence mais absent de notre rendu et *vice versa*). Cette figure montre pour un ensemble de ces valeurs, mesurées sur un ensemble d'images issu du survol de deux jeux de données : le spectrogramme et le Puget Sound. Comme on peut le constater, la qualité de rendu est assez fortement liée au paramètre k qui adapte la distribution de la



(a) $k = 1$ **(b)** $k = 1,25$ **(c)** $k = 1,5$

(d) $k = 1,75$ **(e)** $k = 2$

Figure 8.6 – Influence du paramètre k de l'heuristique. Plus la valeur du paramètre k croît, plus la précision du lancer de rayons augmente. Le paramètre T est fixe ($T = 200$).

précision du lancer de rayons. En effet, quand nous utilisons une valeur basse pour définir k, un plus grand nombre de cellules sont rendues en utilisant un faible niveau de précision. Cela implique nécessairement une dégradation de la qualité globale du rendu. En contre-partie, la vitesse de rendu croît. En ce qui concerne le paramètre T, son impact sur la qualité et la vitesse du rendu est moins important que pour le paramètre k. On peut toutefois noter que la vitesse du rendu décroît lors de l'augmentation de sa valeur, mais la qualité globale du rendu n'est que très légèrement améliorée.

Cependant, on remarque également, et plus spécialement sur la figure 8.8, qu'avec l'algorithme de lancer de rayons, plus de pixels sont manquants sur des jeux de données où les reliefs forment un grand nombre de détails fins mais à hautes fréquences, comme cela est le cas sur le jeu de données représentant le spectrogramme. Cela est plus particulièrement marqué lorsque l'angle de vue est rasant, par rapport au plan de base des données.

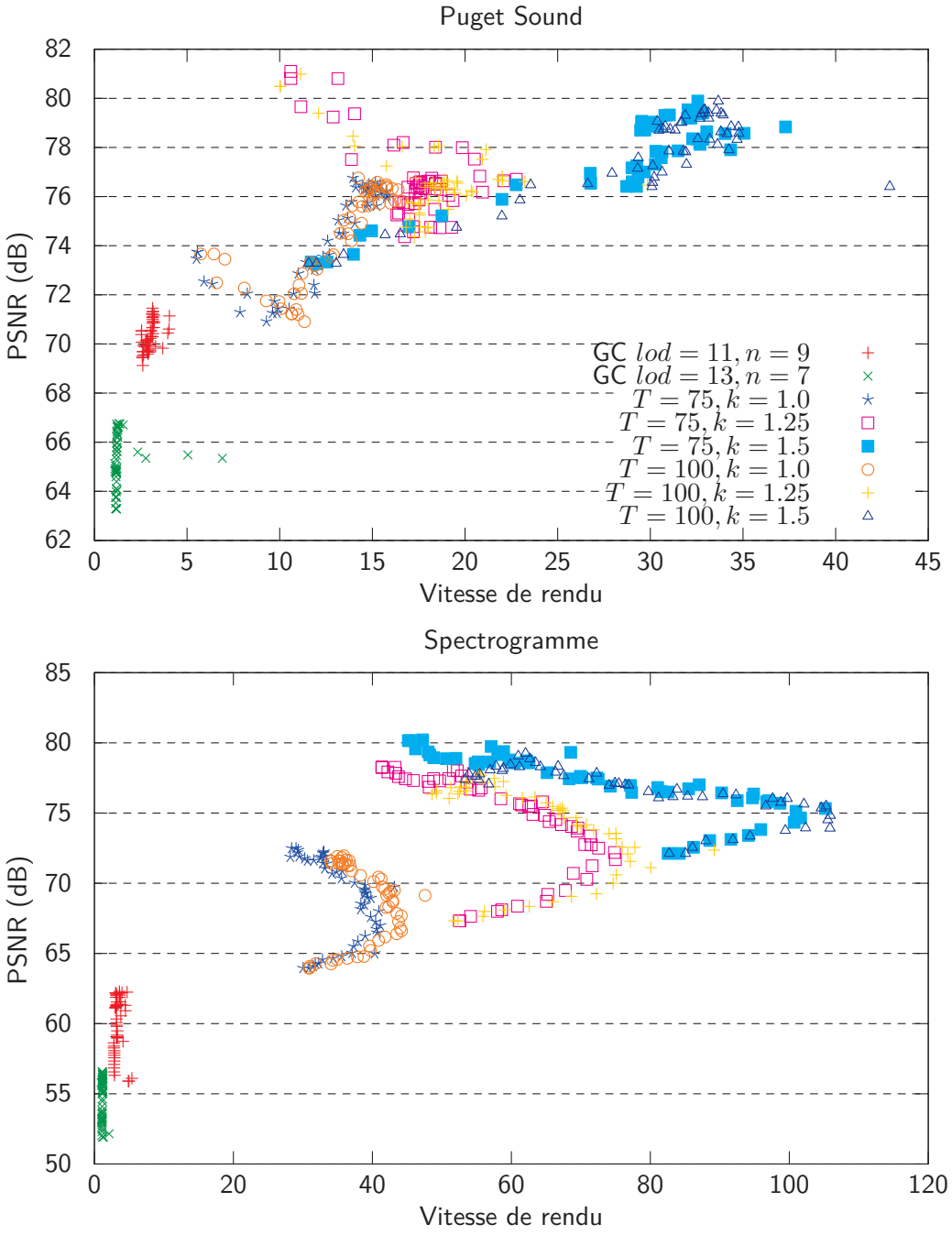


Figure 8.7 – Comparaison, en terme de mesures de PSNR et de vitesse, entre notre méthode et les *geometry clipmaps* (GC). La valeur d'erreur de PSNR (sur l'axe vertical) est exprimée en fonction de la vitesse de rendu (axe horizontal). Chaque point représente la valeur de PSNR et la vitesse de rendu pour une image, issue d'un survol au-dessus du jeu de données représentant le Puget Sound (en haut) et un spectrogramme (en bas).

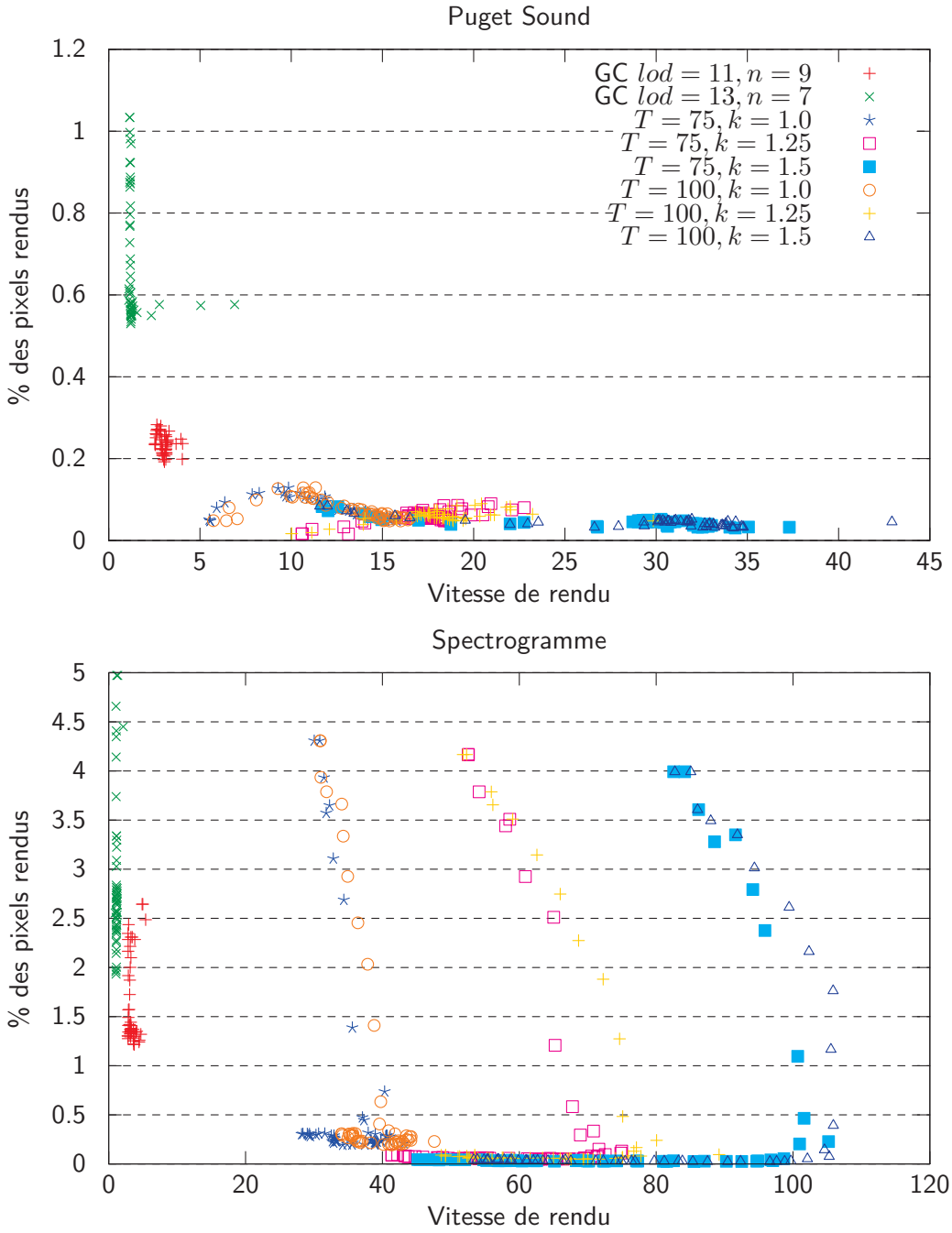


Figure 8.8 – Comparaison, en terme de pixels manquants et de vitesse, entre notre méthode et les *geometry clipmaps* (GC). La valeur exprimée est la proportion de pixels manqués, en fonction de la vitesse de rendu pour le même ensemble d'images. Puget Sound en haut et spectrogramme en bas.

2 Comparaison avec les *Geometry Clipmaps*

Afin d'évaluer la qualité globale de la méthode de visualisation développée, nous avons souhaité la comparer à une méthode existante et faisant référence dans le domaine de la visualisation de terrains. Nous nous sommes orientés vers la technique des *geometry clipmaps* proposée par Losasso et Hoppe [LH04] et décrite à la section 3.2 page 105. Afin d'implanter cette méthode pour la tester, nous avons plus spécialement choisi d'implanter sa version pour processeurs graphiques programmables, décrite par Asirvatham et Hoppe [AH05]. Le choix des *geometry clipmaps* s'est imposé car, à l'heure actuelle, cette techniques est l'une des méthodes faisant référence dans le domaine du rendu de terrains. Elle est parfaitement adaptée aux processeurs graphiques courants, tout en proposant une qualité de rendu de haut niveau. Cependant, cette méthode ne semble pas être celle qui propose la meilleure qualité de rendu. En effet, elle utilise un mécanisme de niveaux de détails qui ne prend pas en compte les données. Par conséquent, les méthodes basées sur des maillages hiérarchiques mais avec des niveaux de détails progressifs fournissent des rendus plus fidèles. En contre-partie, elles se montrent plus compliqués à implanter pour un gain de performances générales qui reste minime.

Implantation et paramètres d'exécution Afin de simplifier son implantation, nous n'avons pas implanter complètement la méthode telle qu'elle est décrite dans l'article original [LH04]. En effet, dans cet article, les auteurs proposent de coupler la méthode de rendu à un mécanisme de compression des données, afin de gérer des masses de données de taille arbitraire. Nous nous sommes limités, dans notre implantation, à des terrains résidant entièrement dans la mémoire graphique, mais en conservant le schéma de simplifications géométriques mis en place par les auteurs. L'implantation a été effectuée en suivant la description donnée par Asirvatham et Hoppe [AH05], c'est-à-dire entièrement sur le processeur graphique. Pour les tests menés et décrits dans cette section, nous avons utilisé les valeurs de paramètres suivants :

- la grille de maillage définie pour un niveau de détails donnés contient un nombre $n = 511$ élévations par côté ;
- le nombre de niveaux de détails est fixé à 11 ;
- la zone de transitions entre chaque niveau de détails, où s'effectue une composition géométrique entre deux niveaux de détails adjacents, a une taille de $n/10$.

Ces différentes valeurs de paramètres sont celles données par les auteurs des différentes implantations et avec lesquelles ils prétendent obtenir les meilleurs résultats. Afin de tester cette méthode sur un rendu plus dégradé, nous avons également effectué certains tests en modifiant le nombre de niveaux de détails (13 niveaux de détails) et la taille des grilles ($n = 127$).

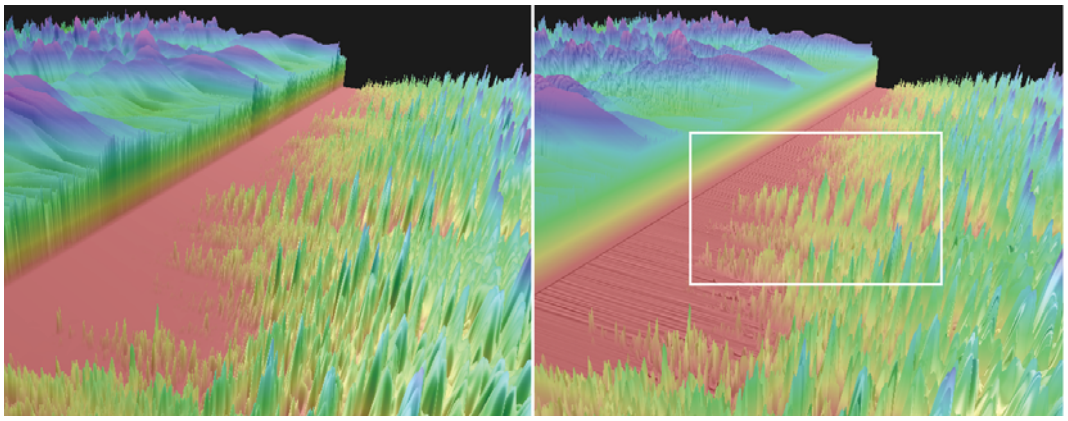
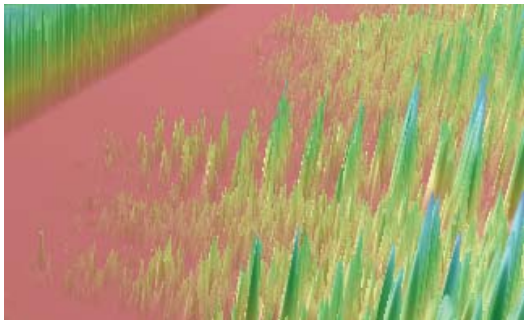


Figure 8.9 – Comparaison du rendu entre notre méthode (à gauche) et un rendu de référence via un maillage à pleine résolution des données (à droite).

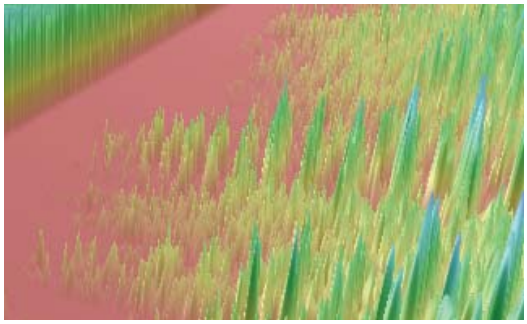
Mesures effectuées et résultats obtenus Nous avons principalement effectué un comparatif des rendus obtenus entre notre méthode et les *geometry clipmaps*, sur un ensemble d'image et sur plusieurs jeux de données tests. Nous nous sommes attachés à mesurer, dans les différents cas, la vitesse du rendu et également à évaluer la qualité de celui-ci, via la mesure d'une valeur de PSNR, telle que décrite précédemment. À chaque fois, nous avons testé notre méthode pour différents réglages de l'heuristique.

La première observation que l'on peut faire est que les *geometry clipmaps* introduisent un lissage des détails fins du relief. Ceci est particulièrement bien visible sur la figure 8.10, où nous avons rendu le spectrogramme via notre méthode, en utilisant une taille de cellules de 64×64 élévations. Ce lissage est dû à l'utilisation, par les *geometry clipmaps*, d'un mécanisme de niveaux de détails uniquement basé sur la distance au point de vue et non pas sur une adaptation aux données visualisées. Ces résultats sont corroborés par les mesures d'erreurs de PSNR effectuées pour la figure 8.7. En effet, notre méthode hybride obtient de meilleurs résultats, en terme de PSNR, que les *geometry clipmaps*. Afin de mettre encore plus en évidence ce lissage, nous avons également mesuré cette erreur pour un réglage plus dégradé des *geometry clipmaps*.

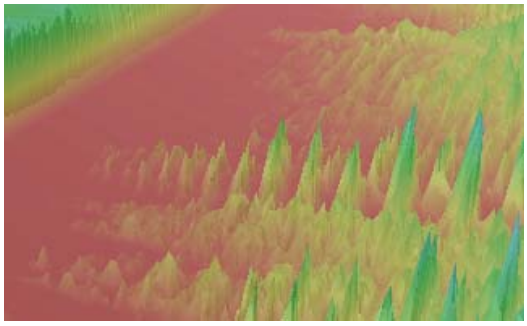
Comme notre heuristique de choix du niveau de précision du rendu prend en compte les données, on constate également que notre méthode conserve mieux les reliefs de la surface du champ de hauteurs, par rapport aux *geometry clipmaps*. En effet, sur des jeux de données présentant des reliefs de grandes amplitudes ou qui forment de multiples détails répartis sur les données avec une assez grande fréquence, on constate que les erreurs au niveau des silhouettes (*i.e.* au niveau des lignes de crêtes) sont assez faibles, alors qu'elles restent



(a) $k = 1,0, T = 100, PSNR \approx 68$



(b) $k = 1,5, T = 100, PSNR \approx 78$



(c) *Geometry clipmaps* : $n = 511$, 11 niveaux de détails, $PSNR \approx 59$

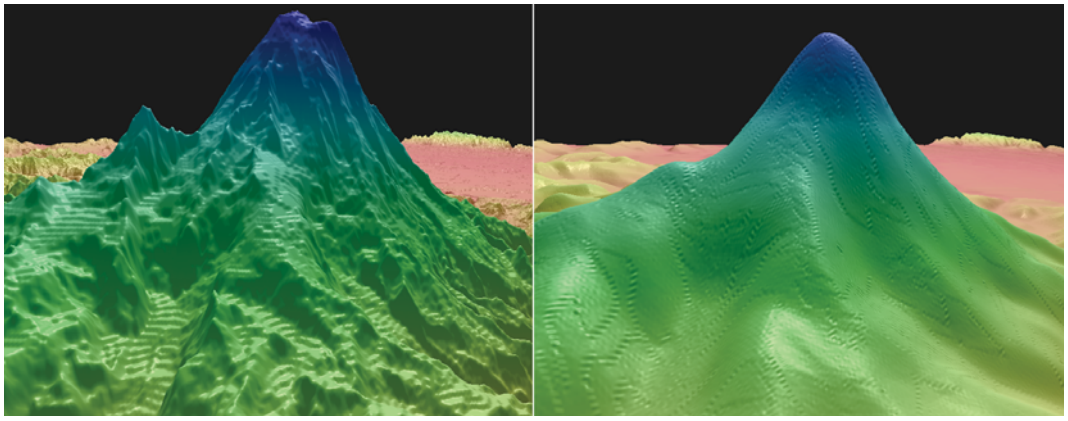
Figure 8.10 – Comparaison visuelle entre la méthode proposée, (a) et (b), et les *geometry clipmaps* (c). Cette figure est un agrandissement d'une zone extraite de la figure 8.9.

plus présentes avec les *geometry clipmaps*, comme cela peut être visible sur la figure 8.10. En fait, en comparant le rendu obtenu à un rendu de référence (par un maillage à pleine résolution des données), comme sur la figure 8.9, on constate qu'un certain nombre de pixels ne sont pas rendus correctement (pas avec la bonne profondeur) ou sont tout simplement absents. Ce phénomène est particulièrement localisé au niveau des lignes de crêtes. Ces observations visuelles sont confirmées par les mesures de PSNR effectuées et données par la figure 8.7. Ces artéfacts de rendu sont plus clairement mis en évidence par les mesures présentées sur la figure 8.8. Comme on peut le constater sur cette figure, en fonction des paramètres de l'heuristique, notre technique de rendu arrive ainsi à minimiser ces erreurs de rendu.

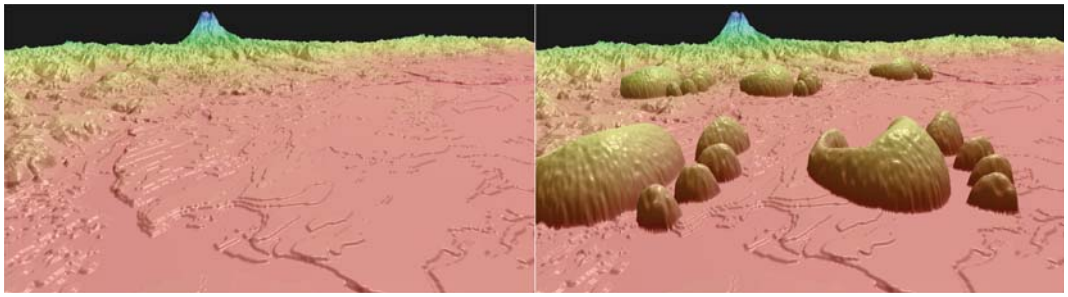
En ce qui concerne la vitesse de rendu, il faut tout de même noter que les *geometry clipmaps* conservent un avantage non négligeable en maintenant une vitesse d'affichage extrêmement élevée, comme on peut le voir sur les figures 8.7 page 144 et 8.8 page 145.

Néanmoins un certains nombre d'expérimentations sur différents jeux de

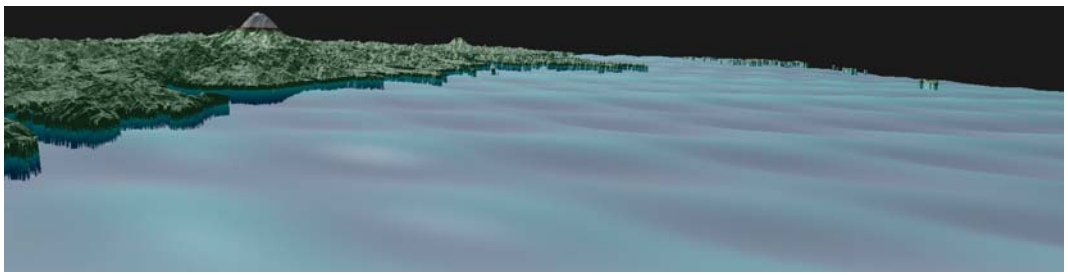
données ont montré que notre méthode se montre avantagée sur des champs de hauteurs où les reliefs présentent une fréquence assez élevée des détails. Dans ce cas, la géométrie est bien préservée, sans phénomène de lissage du rendu, au niveau des détails les plus fins, tout en maintenant une vitesse de rendu pouvant être qualifiée de temps réel.



(a) Simulation d'érosion des données.



(b) Édition locale des données.



(c) Simulation de vagues d'océan.

Figure 8.11 – Illustration du rendu sur de données modifiées dynamiquement. Chaque type de modifications s'effectue en parallèle du rendu.

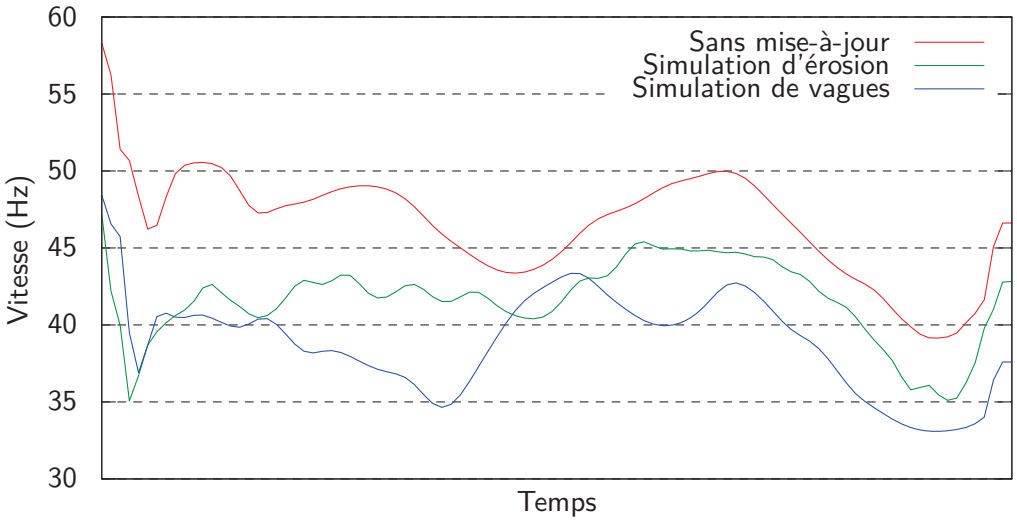


Figure 8.12 – Comparaison des vitesses de rendu durant l'édition des données : sans modification, durant une simulation d'érosion et durant une simulation de vagues d'océan.

3 Données dynamiques

Un des atouts majeurs de notre méthode rendu est son habilité à gérer les jeux de données dynamiques. En effet, comme nous l'avons mentionné précédemment, notre méthode n'introduit aucune dépendance entre les cellules, durant la phase de rendu. Par conséquent, une modification des données reste locale à la zone où celle-ci s'applique, et ne se propage pas à l'ensemble des données visualisées. De même, des modifications globales des données sont également possibles en temps réel, pendant le rendu, à partir du moment où l'algorithme de modification de celles-ci peut s'exécuter suffisamment rapidement. Ainsi, aucune contrainte n'est appliquée aux élévations modifiées, hormis le fait qu'elles sont supposées être comprises entre 0 et 1.

Ainsi comme nous l'avons présenté à la section 5, différents processus de modification des données ont été implantés. Les résultats obtenus par ces mécanismes sont présentés sur la figure 8.11, où le jeu de données représentant la région du Puget Sound a été utilisé. Les différents rendus ont été effectués en parallèle de la modification des données. La fréquence d'affichage varie entre environ 35 et 45 Hz.

En ce qui concerne la vitesse de rendu durant une mise-à-jour des données, comme l'indique la figure 8.12, notre méthode de rendu est capable de fournir des performances de rendu temps réel durant la simulation d'érosion

ou durant la simulation de vagues d'océan.

4 Conclusion

À travers cette seconde partie de cette thèse, nous avons étudié et proposé une nouvelle méthode hybride de rendu de données à deux dimensions et demie. Cette technique a été développée dans l'optique de fournir un bon compromis entre vitesse de rendu, qualité et flexibilité. Cela lui permet, entre autres, d'être applicable à des jeux de données évoluant au cours du temps. Pour bâtir cette méthode, nous avons choisi de combiner deux techniques différentes de rendu, à savoir un rendu par un maillage et un rendu par un algorithme de lancer de rayons, exécuté avec plusieurs niveaux de précision. L'élément clé de cette méthode de rendu, réside dans le mécanisme mis en place pour sélectionner la précision du rendu des différentes zones des données. Pour cela, nous avons proposé une heuristique qui est basée sur différents paramètres dont la distance du point de vue, l'angle de vue et les caractéristiques du terrain. Ce mécanisme permet de maintenir un rendu temps réel des données, tout en gardant un niveau de précision élevé. Cette heuristique s'appuie également sur des paramètres définis par l'utilisateur qui permettent, à partir d'un facteur de qualité, de définir un équilibre entre le rendu par des triangles, où aucune erreur de rendu n'est commise, et le lancer de rayons.

Un des avantages majeur de cette méthode est qu'elle n'a recours à aucune structure de données complexe et également à aucune étape de prétraitements appliquée sur les données. Cela en fait une méthode assez simple à mettre en œuvre, en comparaison des méthodes traditionnellement employées pour visualiser ce type de données. De plus, l'utilisation directe des données du champ de hauteurs, durant la phase de rendu, offre la possibilité de l'utiliser sur des données dynamiques, tout en conservant des performances temps réel pour le rendu. En effet, il est possible d'éditer les données au cours du rendu ou bien d'appliquer des mécanismes plus complexes de mise-à-jour de la carte d'élévations, comme une simulation d'érosion ou de vagues d'océan.

En ce qui concerne la qualité du rendu, nous obtenons des résultats tout-à-fait convaincant lorsque nous la comparons aux méthodes "classiques" de visualisation de terrains, comme les *geometry clipmaps*. Ces résultats sont obtenus grâce à l'utilisation d'un rendu par maillage et d'un lancer de rayons à haute précision pour les reliefs, des données, les plus critiques. Par conséquent, les détails les plus fins du reliefs, ainsi que les lignes de crêtes sont particulièrement bien rendus. Cependant, et comme l'ont montré les travaux précédents du domaine, et en particulier Dick *et al.* [DKW09], le lancer de rayons reste

généralement plus lent qu'un rendu par des hiérarchies de maillages adaptatifs. Cela peut être considéré comme la contrepartie du développement d'une méthode flexible et qui autorise un haut niveau de qualité du rendu. Cependant, ces deux critères restent des critères importants pour un certain nombre d'applications, telles que celles liées à la visualisation de données scientifiques ou à la géomatique. Finalement, notre méthode fournit de meilleurs résultats que les méthodes standards, dans le cas où les données sont formées par de petits reliefs se répétant à fréquence élevée, et en maintenant des performances de rendu temps réel.

La division cellulaire opérée sur les données et la totale indépendance entre les cellules au moment du rendu, fournissent un haut degré de flexibilité à notre méthode. Par conséquent, notre méthode nous autorise à facilement sélectionner les parties du terrain pouvant être rendue ou non. Cela peut être intéressant dans le cas des applications qui doivent rendre différentes parties des données, en utilisant différentes techniques de rendu, pour, par exemple, prendre en compte des zones des données plus complexes ne pouvant pas être représentées par un champ de hauteurs.

Pour le moment notre implantation ne bénéficie pas de mécanisme de chargement dynamique des données pour prendre en compte des jeux de données de taille arbitraire. Mais comme nous utilisons une division cellulaire de celles-ci, le rendu de telle masse de données nous semble possible, en ayant recours à un mécanisme *ad hoc* de chargement des données, tels que celui utilisé par Dick *et al.* [DKW09].

Troisième partie

Conclusion

Chapitre 9

Conclusion générale et perspectives

Ce chapitre est dédié à la conclusion de ce mémoire. Ainsi, à la section 1, un bilan globale des travaux décrits au cours de ce manuscrit est proposé. Pour cela, un bref rappel des résultats obtenus ainsi que des contributions apportées par ces différents travaux, est effectué. Comme les développements menés ont soulevés un certain nombres de questions intéressantes, elles sont présentées à la section 2, qui donne également quelques pistes pour des travaux futurs.

1 Bilan des travaux menés

Les différents travaux menés durant cette thèse et qui sont décrits dans ce mémoire, ont chacun apporté un certain nombre de contributions qui sont détaillées dans cette section, en premier lieu pour la numérisation et la visualisation de tableaux d'art, puis pour la visualisation de terrains.

1.1 Numérisation et visualisation de tableaux

L'objectif des travaux menés dans cette thèse a été d'étudier l'emploi d'une méthode hybride pour la visualisation de données à deux dimensions et demie. Dans la première partie de ces travaux, afin de simplifier le problème initiale de la visualisation de ce type de données, nous avons choisi de travailler avec des données présentant des reliefs ayant de faibles amplitudes et une base planaire. Dans le but de donner un cadre plus concret à cette étude, nous avons choisi de traiter le cas de données réelles, présentant comme spécificité ce type de reliefs. Pour cela, nous nous sommes arrêtés sur des données issues de la

numérisation de peintures d'art.

Ainsi, dans la première partie de ces travaux de thèse, nous avons étudié un protocole permettant la numérisation de tableaux. Son objectif est de fournir des données à une méthode de visualisation temps réel et réaliste, dédiée à ces reliefs particuliers. Pour cela, la numérisation des toiles s'effectue en deux temps. La première étape consiste à acquérir la géométrie de la peinture, via un scanner à lumière structurée. La seconde étape est l'acquisition de l'information colorimétrique, via une acquisition du champ de réflectance de la toile. Concernant le rendu de ces données, il s'effectue en simulant les reliefs les plus faibles, au moyen d'un algorithme de *bump mapping*, et en rendant, via un lancer de rayons exécuté par le processeur graphique, les reliefs qui présentent les plus fortes amplitudes. Ces travaux incluent également une chaîne complète de traitements qui permet d'optimiser les données issues de la numérisation, en vue de leur visualisation : extraction de la géométrie de la toile et génération d'un champ de hauteurs, classification du relief pour séparer les reliefs faibles des plus importants, acquisition et compression de l'information colorimétrique via un modèle de Lafortune encodant l'information de réflectance.

Résultats obtenus Comme nous l'avons décrit dans les chapitres 4 et 5, grâce à cette méthode hybride de visualisation, nous pouvons fournir un rendu de qualité, tout en maintenant des performances de haut niveau. En effet, la combinaison, entre une simple simulation des reliefs les plus petits de la mésostructure des objets et un rendu réel des reliefs plus importants, permet d'accroître sensiblement la vitesse de rendu sans en dégrader la qualité. De plus, l'utilisation d'un modèle analytique de réflectance permet une grande fidélité dans le rendu des couleurs de cette fine structure, tout en incluant certains effets lumineux et, en particulier, les reflets spéculaires qui sont d'une grande importance pour un rendu fidèle des objets. En ce qui concerne le protocole de numérisation de toile, il permet de fournir les données numérisées pour les rendre efficacement avec la méthode de visualisation proposée. Le schéma mémoire retenu pour stocker les données offre une représentation compacte des copies virtuelles des toiles en mémoire et autorise l'utilisation de mécanismes de compression standards d'images pour réduire encore le stockage de ces données ou leur transfert par le réseau.

Contributions Au titre des contributions apportées par ces travaux, nous pouvons citer en premier lieu la prise en compte du relief pour la numérisation et la visualisation des peintures. En effet, jusqu'à présent les méthodes d'acquisition et de rendu de toiles ne prenaient pas en compte cette géométrie durant la phase de rendu. Cependant, une des contributions principales de ces travaux reste la mise au point d'une méthode, dédiée à la visualisation de reliefs de faibles amplitudes, qui combine simulation du relief, par un algorithme

de *bump mapping*, avec un algorithme de lancer de rayons. Cette combinaison s'effectue via un mécanisme adaptatif permettant d'adapter le rendu aux paramètres de visualisation mais également aux données, en désactivant localement le lancer de rayons lorsqu'il n'est pas visuellement nécessaire.

Une autre contribution importante de ces travaux réside dans le modèle retenu pour représenter l'information de couleurs de la peinture. En effet, grâce à l'information de relief stockée dans le champ de hauteurs, il est possible d'intégrer cette information au rendu de la réflectance et ainsi d'améliorer le rendu par rapport à l'utilisation d'une fonction de réflectance bi-directionnelle sous forme d'une fonction de texture bi-directionnelle. Cela permet de découpler l'information de relief de celle de réflectance.

La dernière contribution majeure qui peut être mentionnée, repose sur la chaîne complète de numérisation mise en place pour acquérir la géométrie de la peinture et également la réflectance au moyen d'un scanner à lumière structurée et d'une structure mécanique rigide qui permet de repérer précisément les positions relatives de la caméra et de la source lumineuse par rapport à la toile. Ce protocole de numérisation s'appuie également sur le développement d'un ensemble d'outils simples, nécessaires aux traitements des données acquises.

Ces travaux ont fait l'objet de deux publications dans des conférences d'audience internationale [LAD07b] et nationale [LAD07a].

1.2 Visualisation de terrains

Après avoir étudié le comportement et surtout la possibilité d'utiliser une méthode hybride pour visualiser des données à deux dimensions et demie, mais présentant des reliefs de faibles amplitudes, nos efforts se sont concentrés sur des données présentant une échelle plus grande quant aux reliefs qui les composent. C'est la raison pour laquelle nous nous sommes particulièrement intéressés aux données topographiques.

Dans la seconde partie de cette thèse, nous avons développé une méthode permettant la visualisation temps réel de terrains, avec pour objectif de fournir une méthode efficace et flexible mais offrant un haut niveau de qualité pour la fidélité du rendu des données. Cette méthode a été développée dans la lignée de la méthode proposée pour la visualisation des reliefs de faibles amplitudes. C'est la raison pour laquelle elle est également basée sur une approche hybride. Dans ce cas, le rendu s'effectue en combinant un rendu des reliefs par un maillage et par un algorithme de lancer de rayons. Un mécanisme adaptatif permet ainsi de sélectionner la méthode de rendu la plus adaptée aux conditions

de visualisation courantes et aux reliefs des différentes parties des données, pour optimiser le rendu. Ce mécanisme permet également de gérer différents niveaux de précision pour que la méthode puisse obtenir des performances de rendu temps réel. La flexibilité et la simplicité de la méthode ayant été les principaux objectifs de cette technique de visualisation, elle a été développée pour réduire au maximum les étapes de pré-traitements sur les données, afin de pouvoir gérer les données évoluant dynamiquement au cours du rendu.

Résultats Comme le précise l'analyse détaillée de la méthode proposée, au chapitre 8, cette méthode de rendu de données à deux dimensions et demie fournit des résultats assez intéressants au niveau qualitatif, avec une très bonne préservation des détails des reliefs les plus fins. En effet, elle n'introduit pas d'effets notables de lissages de la surface des données, comme cela peut être le cas avec certaines méthodes classiques de rendu de terrains. Au niveau de la vitesse de rendu et grâce au mécanisme adaptatif, notre méthode est capable de fournir une navigation en temps réel à travers les données. Ces performances temps réels sont également maintenues sur des jeux de données dynamiques qui évoluent ou qui peuvent être modifiées durant le rendu.

Contributions La principale contribution de ces travaux est donc une nouvelle approche hybride, combinant rastérisation et lancer de rayons, pour la visualisation de terrains. Le choix de la méthode de rendu s'effectue via une heuristique, simple à évaluer, permettant à la fois le choix de l'algorithme de rendu et la précision de celui-ci. L'apport de cette heuristique réside dans sa capacité à prendre en compte le relief des données pour déterminer la méthode de rendu à employer ainsi que pour calculer sa précision. Cette approche plus flexible permet, pour un coût nul, la prise en compte de jeux de données dynamiques, tout en conservant des performances de rendu temps réel et une grande qualité de ce dernier. En effet, cette méthode de visualisation ne requiert aucune phase de pré-traitements des données et est capable de travailler directement sur les cartes d'élévations pour les visualiser, sans introduire de structures de données complexes.

Ces travaux ont fait l'objet d'une publication dans une conférence d'audience internationale [AGD10].

2 Limitations et perspectives

Les deux méthodes de visualisation développées connaissent un certain nombre de limitations ou soulèvent plusieurs problèmes, parfois complexes, qu'il convient de détailler. Cependant, certains de ces différents problèmes

peuvent se révéler complexes et donner lieu à des travaux futurs.

2.1 Acquisition et visualisation de tableaux

En ce qui concerne la numérisation de tableaux d'art, notre méthode souffre d'une limitation quant à la taille des toiles que l'on peut numériser. Cette limitation est en particulier vraie pour la phase d'acquisition de la réflectance. En effet, pour cette dernière, nous sommes contraints par la structure mécanique rigide qui est employée pour définir les positions relatives de la caméra et de la source lumineuse par rapport à la toile. En effet, pour la numérisation de la seule géométrie, des techniques permettant de recaler plusieurs images télémétriques entre elles existent [LD06]. Pour ce qui est de la réflectance, comme l'information capturée nécessite de connaître les positions précises de la source lumineuse, de la caméra et de la toile, le problème se révèle être beaucoup plus complexe. Cependant, à notre connaissance, il n'existe pas, à l'heure actuelle, de techniques et de matériels dédiés à l'acquisition de la réflectance, pour de grandes surfaces. Ainsi, il serait intéressant de développer une méthode permettant, à la manière de la géométrie, de diviser l'acquisition en plusieurs fois, puis de recaler les échantillons obtenus entre eux.

Au niveau de la méthode de visualisation proprement dite, là aussi certaines améliorations sont possibles. L'accroissement de la vitesse de rendu fait partie de ces améliorations envisageables. En effet, bien qu'elle soit déjà d'un haut niveau, il reste toujours possible d'améliorer encore les performances. Actuellement, aucun mécanisme n'est utilisé pour accroître la vitesse d'exécution du lancer de rayons. Or, il existe des techniques très efficaces permettant ce type d'optimisation. Dans ce cas précis, on pourrait imaginer l'ajout d'un mécanisme calqué sur les *maximum mipmaps* [TIS08]. Un tel ajout permettrait également de gagner en précision pour le rendu des reliefs les plus importants de la toile, en évitant les artéfacts qui peuvent apparaître avec l'algorithme de lancer de rayons que nous employons, lorsque celui-ci est utilisé avec une précision trop dégradée.

Une limitation importante de notre méthode de visualisation de reliefs de faibles amplitudes est liée à la classe des objets qui sont pris en compte. En effet, seuls les objets à base planaire avec une mésostructure formée de faibles reliefs par rapport à la macrostructure peuvent être visualisés. Une extension naturelle est donc d'élargir la méthode à la prise en compte d'objets dont la macrostructure n'est plus simplement modélisable par un plan. L'atout de cette extension serait ainsi de maintenir une prise en compte du relief de la mésostructure, pour moduler la précision du rendu (simulation par *bump mapping* ou non) des différentes parties de la mésostructure.

2.2 Visualisation de terrains

Pour la visualisation de terrains, un certain nombre de limitations touche la méthode développée et sont liées à certains problèmes complexes. Par conséquent, notre technique de rendu peut soulever quelques interrogations quant à ces différents points.

La première limitation qui peut être mise en évidence, est inhérente à l'implantation que nous avons faite de cette technique de visualisation. En effet, dans notre implantation, il est nécessaire que toutes les données, représentant le terrain, soient présentes en mémoire graphique, durant le rendu. Ceci pose une limitation quant à la taille des données que l'on peut visualiser. Cependant, comme les données sont divisées suivant un schéma régulier de cellules, il est possible d'utiliser un mécanisme de chargement dynamique des données utilisant une telle division des données, comme, par exemple, dans le cas de Dick *et al.* [DKW09].

Afin de préserver au mieux les reliefs fins des données, nous avons choisi de ne pas avoir recours à une structure hiérarchique appliquée sur les données géométriques. Ce choix, bien que présentant certains avantages qui sont présentés plus en détails dans les chapitres précédents, introduit différentes limitations liées, en particulier, à l'échantillonnage régulier qui caractérise les données à deux dimensions et demie. En effet, lors du rendu de ces données, en particulier pour les zones lointaines, par rapport au point de vue, se produit un sur-échantillonnage des données, par rapport au nombre de pixels rendus (*i.e.* plusieurs échantillons se projettent à l'écran sur un seul pixel). Cela se traduit visuellement par des effets d'aliasing et de moirées qui peuvent survenir, surtout lors des déplacements de la caméra. La solution habituellement employée, pour limiter ce genre de problèmes, consiste à utiliser une structure hiérarchique sur la géométrie. Cependant, ce type de structures introduit généralement un lissage, parfois excessif, des données, comme on peut le voir sur les *geometry clipmaps*. De plus, la génération de ce type de structures est particulièrement coûteuse en calculs, ce qui limite son application à des données statiques. Par conséquent, dans notre cas, cette méthode ne semble pas être la solution adéquate. Une meilleure solution est sans doute à rechercher en utilisant une méthode basée sur un procédé par sur-échantillonnage du rendu. Dans ce cas de figure, l'algorithme de lancer de rayons semble assez bien adapté. Toutefois, ce type de procédés est extrêmement coûteux en temps de calculs, ce qui pénaliserait inévitablement la vitesse de rendu.

Par conséquent, une des améliorations nécessaires à cette méthode est d'accroître encore la vitesse de rendu, mais bien évidemment sans en dégrader

la qualité ou la flexibilité. Pour cela, plusieurs pistes semblent être envisageables. La première consiste à développer un système hiérarchique, non pas basé sur les données, mais sur les différentes cellules, ce qui permettrait de regrouper celles nécessitant moins de précision. Cela réduirait également le nombre de pixels évalués plusieurs fois de suite, pour rendre une même image. Parallèlement à cette amélioration, la passe de rendu par le lancer de rayons doit pouvoir être accélérée. En effet, l'utilisation de mécanismes, dans l'esprit de *maximum mipmaps* de Tevs *et al.* [TIS08], peut être appliquée au lancer de rayons que nous employons. Sur ces deux points, il faut toutefois être vigilent quant à la préservation de la flexibilité de la méthode, et notamment sur la possibilité de gérer des jeux de données dynamiques, ainsi que sur la préservation du haut niveau de qualité que nous obtenons, en évitant d'introduire un lissage des données, en particulier sur les reliefs les plus fins.

2.3 Perspectives globales

Au cours de cette thèse, nous avons présenté une méthode permettant la visualisation de données à deux dimensions et demie. Un des principaux champs d'applications de ces données est la visualisation de terrains. Cependant, il peut être pertinent de s'interroger sur le type de terrains représentables via ce format de données. La classe des terrains que l'on peut modéliser en utilisant ces données, bien que très fréquemment employées, restent assez simples. Des travaux récents [PEGMG09], ont montré qu'en utilisant des structures de données adaptées (et plus seulement limitées à deux dimensions et demie), il devient possible de modéliser des terrains beaucoup plus complexes, bien que leur rendu, en temps réel, reste un problème ouvert. Une question peut ainsi se poser. Est-il possible, à partir d'une méthode dont l'approche est similaire à la notre, de rendre ce type de terrains, en modifiant, par exemple, certaines zones d'un champ de hauteurs rendu.

Une solution à ce problème pourrait être apportée par les récentes avancées des matériels graphiques. En effet, à l'heure où ces lignes sont écrites, de nouvelles possibilités, notamment au niveau de la subdivision dynamique de maillages, sont implantées au niveau des cartes graphiques. Cette capacité pourrait, peut-être, être exploitée à ces fins.

Annexes

Bibliographie

[AGBL07] D. Akça, A. Gruen, B. Breuckmann et C. Lahanier : High definition 3D-scanning of arts objects and paintings. *In Optical 3-D Measurement Techniques VIII*, pages 55–58. Chair of Photogrammetry and Remote Sensing, Institute of Geodesy and Photogrammetry, ETH Zürich, juillet 2007.

[AGD10] Lucas Ammann, Olivier Génevaux et Jean-Michel Dischler : Hybrid rendering of dynamic heightfields using ray-casting and mesh rasterization. *In Proceedings of Graphics Interface 2010*, pages 161–168. Canadian Human-Computer Communication Society, may 2010.

[AH05] Arul Asirvatham et Hugues Hoppe : *GPU Gems 2*, chapitre 2. Terrain Rendering Using GPU-Based Geometry Clipmaps. Addison-Wesley Professional, 2005.

[AW87] John Amanatides et Andrew Woo : A fast voxel traversal algorithm for ray tracing. *In* Guy Maréchal, éditeur : *Proceedings of the European Computer Graphics Conference and Exhibition*, pages 3–10, Amsterdam, août 1987. North-Holland.

[BAD+01] M. Bóo, M. Amor, M. Doggett, J. Hirche et W. Strasser : Hardware support for adaptive subdivision surface rendering. *In HWWS '01 : Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 33–40, New York, NY, USA, 2001. ACM.

[BAV98] L. Balmelli, S. Ayer et M. Vetterli : Efficient algorithms for embedded rendering of terrain models. *In Proceedings IEEE International Conference on Image*, pages 914–918, 1998.

[BD06] Lionel Baboud et Xavier Décoret : Rendering geometry with relief textures. *In GI '06 : Proceedings of the 2006 conference on Graphics interface*, pages 195–201, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society.

[BDHK99] Konstantin Baumann, Juergen Doellner, Klaus Hinrichs et Oliver Kersting : A hybrid, hierarchical data structure for real-time terrain visualization. *Computer Graphics International Conference*, 0:85, 1999.

[BDZ06] D.A. Balciunas, L.P. Dulley et M.K. Zuffo : GPU-assisted ray casting of large scenes. *In IEEE Symposium on Interactive Ray Tracing 2006*, pages 95–103, septembre 2006.

[BGP09] Jonas Bösch, Prashant Goswami et Renato Pajarola : RASTeR : Simple and efficient terrain rendering on the GPU. *In Proceedings of Eurographics 2009 - Areas Papers*, pages 35–42, 2009.

[Bli78] James F. Blinn : Simulation of wrinkled surfaces. *In SIGGRAPH '78 : Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 286–292, New York, NY, USA, 1978. ACM Press.

[BM93] Barry G. Becker et Nelson L. Max : Smooth transitions between bump rendering algorithms. *In SIGGRAPH '93 : Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 183–190, New York, NY, USA, 1993. ACM Press.

[BN76] James F. Blinn et Martin E. Newell : Texture and reflection in computer generated images. *Commun. ACM*, 19(10):542–547, 1976.

[BNH10] Eric Bruneton, Fabrice Neyret et Nicolas Holzschuch : Real-time realistic ocean lighting using seamless transitions from geometry to BRDF. *Computer Graphics Forum*, 29(2), 2010.

[BS05] Tamy Boubekeur et Christophe Schlick : Generic mesh refinement on GPU. *In HWWS '05 : Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 99–104, New York, NY, USA, 2005. ACM.

[BT04] Zoe Brawley et Natalya Tatarchuk : Parallax occlusion mapping : Self-shadowing, perspective-correct bump mapping using reverse height map tracing. *In ShaderX3 : Advanced Rendering with DirectX and OpenGL*, chapitre 2 - Rendering Techniques. Charles River Media, 2004.

[CCC87] Robert L. Cook, Loren Carpenter et Edwin Catmull : The reyes image rendering architecture. *In SIGGRAPH '87 : Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 95–102, New York, NY, USA, 1987. ACM.

[CD03] Eric Chan et Frédo Durand : Rendering fake soft shadows with smoothies. *In Proceedings of the 14th Eurographics workshop on Rendering*, pages 208–218, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[CGG+03a] Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio et Roberto Scopigno : BDAM - batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum*, 22(3):505–514, 2003.

[CGG+03b] Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio et Roberto Scopigno : Planet-sized batched dynamic adaptive meshes (P-BDAM). *In VIS '03 : Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 20, Washington, DC, USA, 2003. IEEE Computer Society.

[CN01] Baoquan Chen et Minh Xuan Nguyen : POP : a hybrid point and polygon rendering system for large data. *In VIS '01 : Proceedings of the conference on Visualization '01*, pages 45–52, Washington, DC, USA, 2001. IEEE Computer Society.

[COL96] Daniel Cohen-Or et Yishay Levanoni : Temporal continuity of levels of detail in delaunay triangulated terrain. *In VIS '96 : Proceedings of the 7th conference on Visualization '96*, pages 37–42, Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[Coo84] Robert L. Cook : Shade trees. *SIGGRAPH Comput. Graph.*, 18(3):223–231, 1984.

[CORLS96] Daniel Cohen-Or, Eran Rich, Uri Lerner et Victor Shenkar : A real-time photo-realistic visual flythrough. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):255–265, 1996.

[CPS97] P. Cignoni, E. Puppo et R. Scopigno : Representation and visualization of terrain surfaces at variable resolution. *The Visual Computer*, 13(5):199–217, 1997.

[CS93] Daniel Cohen et Amit Shaked : Photo-realistic imaging of digital terrains. *Computer Graphics Forum (Eurographics '93)*, 12(3):363–373, 1993. Held in Oxford, UK.

[CT81] Robert L. Cook et Kenneth E. Torrance : A reflectance model for computer graphics. *In SIGGRAPH '81 : Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, pages 307–316, New York, NY, USA, 1981. ACM.

[Dan80] Per-Erik Danielsson : Euclidean distance mapping. *Computer Graphics Image Processing*, 14(3):227–248, novembre 1980.

[DFKP05] Leila De Floriani, Leif Kobbelt et Enrico Puppo : A survey on data structures for level-of-detail models. *In Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 49–74. Springer Berlin Heidelberg, 2005.

[DGNK99] Kristin J. Dana, Bram Van Ginneken, Shree K. Nayar et Jan J. Koenderink : Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, 1999.

[DH00] Michael Doggett et Johannes Hirche : Adaptive view dependent tessellation of displacement maps. *In HWWS '00 : Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 59–66, New York, NY, USA, 2000. ACM Press.

[Die00] Sim Dietrich : Elevation maps. Rapport technique, NVIDIA Corporation, 2000.

[Dis98] Jean-Michel Dischler : Efficient rendering macro geometric surface structures with bi-directional texture functions. *In* George Drettakis et Nelson Max, éditeurs : *Rendering Techniques '98*, Eurographics, pages 169–180. Springer-Verlag Wien New York, 1998.

[DKS01] Michael C. Doggett, Anders Kugler et Wolfgang Strasser : Displacement mapping using scan conversion hardware architectures. *Computer Graphic Forum*, 20(1):13–26, 2001.

[DKW09] Christian Dick, Jens Krüger et Rüdiger Westermann : GPU ray-casting for scalable terrain rendering. *In Proceedings of Eu-*

rographics 2009 - Areas Papers, pages 43–50, 2009.

[DNGK97] Kristin J. Dana, Shree K. Nayar, Bram Van Ginneken et Jan J. Koenderink : Reflectance and texture of real-world surfaces authors. *In CVPR '97 : Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 151, Washington, DC, USA, 1997. IEEE Computer Society.

[Don05] William Donnelly : Per-pixel displacement mapping with distance functions. *In GPU Gems 2*, chapitre 8. Addison-Wesley Educational Publishers Inc, 2005.

[DS04] Carsten Dachsbacher et Marc Stamminger : Rendering procedural terrain by geometry image warping. *In* Alexander Keller et Henrik Wann Jensen, éditeurs : *Eurographics Symposium on Rendering*, pages 103–110, Norrköping, Sweden, 2004. Eurographics Association.

[DSW09] Christian Dick, Jens Schneider et Rüdiger Westermann : Efficient geometry compression for GPU-based decoding in real-time terrain rendering. *Computer Graphic Forum*, 28(1):67–83, 2009.

[Dum06] Jonathan Dummer : Cone step mapping. `http://lonesock.net/files/ConeStepMapping.pdf`, 2006.

[DWS+97] Mark Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Miller, Charles Aldrich et Mark B. Mineev-Weinstein : ROAMing terrain : real-time optimally adapting meshes. *In VIS '97 : Proceedings of the 8th conference on Visualization '97*, pages 81–88, Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[EKT01] W. Evans, D. Kirkpatrick et G. Townsend : Right-triangulated irregular networks. *Algorithmica*, 30(2):264–286, 2001.

[FGG+05] Raffaella Fontana, Maria Chiara Gambino, Marinella Greco, Luciano Marras, Enrico M. Pampaloni, Anna Pelagotti, Luca Pezzati et Pasquale Poggi : 2D imaging and 3D sensing data acquisition and mutual registration for painting conservation. *In* J.-Angelo Beraldin, Sabry F. El-Hakim, Armin Gruen et James S. Walton, éditeurs : *Videometrics VIII*, volume 5665, pages 51–58. SPIE, 2005.

[FL79] Robert J. Fowler et James J. Little : Automatic extraction of irregular network digital terrain models. *In SIGGRAPH '79 : Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, pages 199–207, New York, NY, USA, 1979. ACM.

[FZPM93] John S. Falby, Michael J. Zyda, David R. Pratt et Randy L. Mackey : NPSNET : Hierarchical data structures for real-time three-dimensional visual simulation. *Computers & Graphics*, 17(1):65–70, 1993.

[GAL03] Gabriele Guidi, Carlo Atzeni et Sara Lazzari : 3d optical scanning diagnostics for Leonardo Da Vinci's "Adorazione dei Magi" conservation. *In 3DIM*, pages 110–115. IEEE Computer Society, 2003.

[GGH02] Xianfeng Gu, Steven J. Gortler et Hugues Hoppe : Geometry images. *ACM Transactions on Graphics*, 21(3):355–361, 2002.

[GH99] Stefan Gumhold et Tobias Hüttner : Multiresolution rendering with displacement mapping. *In* ACM, éditeur : *SIGGRAPH '99. Proceedings 1999 Eurographics/SIGGRAPH workshop on Graphics hardware, Aug. 8–9, 1999, Los Angeles, CA*, pages 55–66, pub-ACM :adr, 1999. ACM Press.

[GMC+06] Enrico Gobbetti, Fabio Marton, Paolo Cignoni, Marco Di Benedetto et Fabio Ganovelli : C-BDAM – compressed batched dynamic adaptive meshes for terrain rendering. *Computer Graphics Forum*, 25(3):333–342, septembre 2006. Proceedings of Eurographics 2006.

[GS03] Paolo Grattoni et Massimiliano Spertino : A mosaicing approach for the acquisition and representation of 3D painted surfaces for conservation and restoration purposes. *Machine Vision and Applications*, 15(1):1–10, octobre 2003.

[GW07] Markus Giegl et Michael Wimmer : Unpopping : Solving the image-space blend problem for smooth discrete lod transitions. *Computer Graphics Forum*, 26(1):46–49, 2007.

[HDJ05] Lok M. Hwa, Mark A. Duchaineau et Kenneth I. Joy : Real-time optimal adaptation for planetary geometry and texture : 4-8 tile hierarchies. *IEEE Transactions on Visualization and Computer Graphics*, 11:355–368, 2005.

[HDKS00] Wolfgang Heidrich, Katja Daubert, Jan Kautz et Hans-Peter Seidel : Illuminating micro geometry based on precomputed visibility. *In SIGGRAPH '00, Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 455–464, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[HEGD04] Johannes Hirche, Alexander Ehlert, Stefan Guthe et Michael Doggett : Hardware accelerated per-pixel displacement mapping. *In GI '04 : Proceedings of Graphics Interface 2004*, pages 153–158, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[HNC02] Damien Hinsinger, Fabrice Neyret et Marie-Paule Cani : Interactive animation of ocean waves. *In ACM-SIGGRAPH/EG Symposium on Computer Animation, SCA 2002, July, 2002*, pages 161–166, San Antonio, Texas, Etats-Unis, juillet 2002. ACM Press.

[Hop96] Hugues Hoppe : Progressive meshes. *In SIGGRAPH '96 : Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108, New York, NY, USA, 1996. ACM.

[Hop98] Hugues Hoppe : Smooth view-dependent level-of-detail control and its application to terrain rendering. *In VIS '98 : Proceedings of the conference on Visualization '98*, pages 35–42, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[HS98] Wolfgang Heidrich et Hans-Peter Seidel : Ray-tracing procedural displacement shaders. *In Proceedings of the Graphics Interface 1998 (GI-98)*, pages 8–16, 1998.

[HS01] Ziyad S. Hakura et John M. Snyder : Realistic reflections and refractions on graphics hardware with hybrid rendering and layered environment maps. *In Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 289–300, London, UK, 2001. Springer-Verlag.

[HS04] Christian Henning et Peter Stephenson : Accelerating the ray tracing of height fields. *In GRAPHITE '04 : Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 254–258, New York, NY, USA, 2004. ACM Press.

[HTSG91] Xiao D. He, Kenneth E. Torrance, François X. Sillion et Donald P. Greenberg : A comprehensive physical model for light reflection. *In SIGGRAPH '91 : Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 175–186, New York, NY, USA, 1991. ACM.

[JLLW04] Junfeng Ji, Sheng Li, Xuehui Liu et Enhua Wu : P-Quadtrees : A point and polygon hybrid multi-resolution rendering approach. *In Computer Graphics International Conference*, pages 382–385, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[JMLH01] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy et Pat Hanrahan : A practical model for subsurface light transport. *In SIGGRAPH '01 : Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 511–518, New York, NY, USA, 2001. ACM.

[Joh04] C Johanson : Real-time water rendering - introducing the projected grid concept. Mémoire de Master, Department of Computer Science, Lund University, 2004.

[KE09] Murat Kurt et Dave Edwards : A survey of BRDF models for computer graphics. *SIGGRAPH Computer Graphics*, 43(2):1–7, 2009.

[KK89] J. T. Kajiya et T. L. Kay : Rendering fur with three dimensional textures. *In SIGGRAPH '89 : Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 271–280, New York, NY, USA, 1989. ACM.

[KRS05] Andreas Kolb et Christof Rezk-Salama : Efficient empty space skipping for per-pixel displacement mapping. *In Proc. Vision, Modeling and Visualization (VMV)*, pages 407–414, 2005.

[KS01] Jan Kautz et Hans-Peter Seidel : Hardware accelerated displacement mapping for image based rendering. *In Proceedings of the Graphics Interface 2001 (GI-01)*, pages 61–70, juin 2001.

[KTI+01] Tomomichi Kaneko, Toshiyuki Takahei, Masahiko Inami, Naoki Kawakami, Yasuyuki Yanagida, Taro Maeda et Susumu Tachi : Detailed shape representation with parallax mapping. *In ICAT 2001 : Proceedings of the International Conference on Artificial Reality and Telexistence*, pages 205–208, 2001.

[LAD07a] Frédéric Larue, Lucas Ammann et Jean-Michel Dischler : Chaîne de traitement pour la numérisation et le rendu réaliste

de peintures d'art. *In Journées AFIG 2007*, Marne-la-Vallée, novembre 2007.

[LAD07b] Frédéric Larue, Lucas Ammann et Jean-Michel Dischler : A pipeline for the digitisation and the realistic rendering of paintings. *In 8th International Symposium on Virtual Reality, Archaeology and Cultural Heritage*. Eurographics, november 2007.

[Lar08] Frédéric Larue : *Numérisation de pièces d'art en termes de forme et d'apparence pour la visualisation réaliste en synthèse d'images*. Thèse de doctorat, Université Louis Pasteur de Strasbourg, novembre 2008.

[LD06] Frédéric Larue et Jean-Michel Dischler : Automatic registration and calibration for efficient surface light field acquisition. *In 7th VAST International Symposium on Virtual Reality, Archeology and Cultural Heritage*, pages 171–178. Eurographics, 2006.

[Lev90] Marc Levoy : Efficient ray tracing of volume data. *ACM Trans. Graph.*, 9(3):245–261, 1990.

[Lev02] Joshua Levenberg : Fast view-dependent level-of-detail rendering using cached geometry. *In VIS '02 : Proceedings of the conference on Visualization '02*, pages 259–266, Washington, DC, USA, 2002. IEEE Computer Society.

[LFTG97] Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance et Donald P. Greenberg : Non-linear approximation of reflectance functions. *In SIGGRAPH '97 : Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 117–126, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[LH04] Frank Losasso et Hugues Hoppe : Geometry clipmaps : terrain rendering using nested regular grids. *In SIGGRAPH '04 : ACM SIGGRAPH 2004 Papers*, pages 769–776, New York, NY, USA, 2004. ACM Press.

[LKG+03] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich et Hans-Peter Seidel : Image-based reconstruction of spatial appearance and geometric detail. *ACM Transactions on Graphics*, 22(2):234–257, 2003.

[LKR+96] Peter Lindstrom, David Koller, William Ribarsky, Larry F. Hodges, Nick Faust et Gregory A. Turner : Real-time, continuous level of detail rendering of height fields. *In SIGGRAPH*

'96 : Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pages 109–118, New York, NY, USA, 1996. ACM Press.

[LP01] Peter Lindstrom et Valerio Pascucci : Visualization of large terrains made easy. *In VIS '01 : Proceedings of the conference on Visualization '01*, pages 363–371, Washington, DC, USA, 2001. IEEE Computer Society.

[LP02] Peter Lindstrom et Valerio Pascucci : Terrain simplification simplified : A general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):239–254, 2002.

[LPC+00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade et Duane Fulk : The digital Michelangelo project : 3D scanning of large statues. *In SIGGRAPH '00 : Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 131–144, New York, NY, USA, juillet 2000. ACM Press/Addison-Wesley Publishing Co.

[LPT03] Roberto Lario, Renato Pajarola et Francisco Tirado : HyperBlock-QuadTIN : Hyper-block quadtree based triangulated irregular networks. *In Proceedings IASTED International Conference on Visualization, Imaging and Image Processing (VIIP)*, pages 733–738, 2003.

[LS95] Cheol-Hi Lee et Yeong Gil Shin : An efficient ray tracing method for terrain rendering. *In Pacific Graphics '95*, août 1995.

[LSGS08] Y. Livny, N. Sokolovsky, T. Grinshpoun et J. El Sana : A GPU persistent grid mapping for terrain rendering. *The Visual Computer*, 24(2):139–153, février 2008.

[Max88] Nelson L. Max : Horizon mapping : shadows for bump-mapped surfaces. *The Visual Computer*, 4(2):109–117, 1988.

[McM97] Leonard McMillan, Jr. : *An image-based approach to three-dimensional computer graphics*. Thèse de doctorat, Chapel Hill, NC, USA, 1997.

[MCSP02] K. Martinez, J. Cupitt, D. Saunders et R. Pillay : Ten years of art imaging research. *Proceedings of the IEEE*, 90(1):28 –41, janvier 2002.

[Miy90] Kazunori MIYATA : A method of generating stone wall patterns. *In SIGGRAPH '90 : Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 387–394, New York, NY, USA, 1990. ACM.

[MJ06] Stephan MANTLER et Stefan JESCHKE : Interactive landscape visualization using GPU ray casting. *In GRAPHITE '06 : Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 117–126, New York, NY, USA, 2006. ACM Press.

[MLH02] David K. MCALLISTER, Anselmo LASTRA et Wolfgang HEIDRICH : Efficient rendering of spatial bi-directional reflectance distribution functions. *In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 79–88, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[MM02] Kevin MOULE et Michael D. MCCOOL : Efficient bounded adaptive tessellation of displacement maps. *In Proc. Graphics Interface*, pages 171–180, mai 2002.

[MM05] Morgan MCGUIRE et Max MCGUIRE : Steep parallax mapping. Rapport technique, Report at Brown University, 2005. I3DG Poster.

[MMK03] J. MESETH, G. MÜLLER et R. KLEIN : Preserving realism in real-time rendering of bidirectional texture functions. *In OpenSG Symposium*, pages 89–96, mars 2003.

[MMS+05] G. MULLER, J. MESETH, M. SATTLER, R. SARLETTE et R. KLEIN : Acquisition, synthesis, and rendering of bidirectional texture functions. *Computer Graphics Forum*, 24(1):83–109, 2005.

[MN98] Alexandre MEYER et Fabrice NEYRET : Interactive volumetric textures. *In* George DRETTAKIS et Nelson MAX, éditeurs : *Rendering Techniques '98*, Eurographics, pages 157–168. Springer-Verlag Wien New York, 1998.

[Mus88] F. Kenton MUSGRAVE : Grid tracing : Fast ray tracing for height fields. Technical Report 639, Yale University Dept. of Computer Science Research, 1988.

[Mus94] F. Kenton MUSGRAVE : *Texturing and Modeling*, chapitre 16 : Procedural Fractal Terrains, 17 : QAEB Rendering For Procedu-

ral Models. Morgan Kaufmann Publishers Inc, 1994.

[MW08] Morgan McGuire et Kyle Whitson : Indirection mapping for quasi-conformal relief texturing. *In I3D '08 : Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 191–198, New York, NY, USA, 2008. ACM.

[MWL+99] Stephen R. Marschner, Stephen H. Westin, Eric P. Lafortune, Kenneth E. Torrance et Donald P. Greenberg : Image-based BRDF measurement including human skin. *In* Dani Lischinski et Gregory Ward Larson, éditeurs : *Rendering Techniques '99, Proceedings of the Eurographics Workshop on Rendering*, pages 131–144. Springer, 1999.

[NDM05] Addy Ngan, Frédo Durand et Wojciech Matusik : Experimental analysis of BRDF models. *In Eurographics Symposium on Rendering*, pages 117–126, Konstanz, Germany, 2005. Eurographics Association.

[OBM00] Manuel M. Oliveira, Gary Bishop et David McAllister : Relief texture mapping. *In SIGGRAPH '00 : Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 359–368, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[OKL06] Kyoungsu Oh, Hyunwoo Ki et Cheol-Hi Lee : Pyramidal displacement mapping : a gpu based artifacts-free ray tracing through an image pyramid. *In VRST '06 : Proceedings of the ACM symposium on Virtual reality software and technology*, pages 75–82, New York, NY, USA, 2006. ACM Press.

[OKS03] Marc Olano, Bob Kuehne et Maryann Simmons : Automatic shader level of detail. *In HWWS '03 : Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 7–14, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[Ols04] Jacob Olsen : Realtime procedural terrain generation. October 2004.

[OP05] Manuel M. Oliveira et Fábio Policarpo : An efficient representation for surface details. Rapport technique RP-351, UFRGS, 2005.

[PAC97] Mark Peercy, John Airey et Brian Cabral : Efficient bump mapping hardware. *In SIGGRAPH '97 : Proceedings of the*

24th annual conference on Computer graphics and interactive techniques, pages 303–306, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[Paj98] Renato Pajarola : Large scale terrain visualization using the restricted quadtree triangulation. *In Proceedings Visualization 98*, pages 19–26,515, Los Alamitos, California, 1998. IEEE, Computer Society Press.

[PAL02] Renato Pajarola, Marc Antonijuan et Roberto Lario : QuadTIN : quadtree based triangulated irregular networks. *In VIS '02 : Proceedings of the conference on Visualization '02*, pages 395–402, Washington, DC, USA, 2002. IEEE Computer Society.

[PBMH02] Timothy J. Purcell, Ian Buck, William R. Mark et Pat Hanrahan : Ray tracing on programmable graphics hardware. *ACM Transactions on Graphics*, 21:703–712, juillet 2002.

[PEGMG09] Adrien Peytavie, Éric Galin, Stephane Merillou et Jérome Grosjean : Arches : a framework for modeling complex terrains. *Computer Graphics Forum*, 28:457–467, 2009.

[PFLM78] Thomas K. Peucker, Robert J. Fowler, James J. Little et David M. Mark : The triangulated irregular network. *In Proceedings of the Digital Terrain Models (DTM) Symposium*, pages 516–540, St. Louis, mai 1978. American Society of Photogrammetry.

[PG07] Renato Pajarola et Enrico Gobbetti : Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer : International Journal of Computer Graphics*, 23(8):583–605, 2007.

[PH96] Matt Pharr et Pat Hanrahan : Geometry caching for ray-tracing displacement maps. *In Proceedings of the eurographics workshop on Rendering techniques '96*, pages 31–ff., London, UK, 1996. Springer-Verlag.

[PHL91] J. W. Patterson, S. G. Hoggar et J. R. Logie : Inverse displacement mapping. *Computer Graphics Forum*, 10(2):129–139, juin 1991.

[PO06] Fábio Policarpo et Manuel M. Oliveira : Relief mapping of non-height-field surface details. *In SI3D '06 : Proceedings of the*

2006 symposium on Interactive 3D graphics and games, pages 55–62, New York, NY, USA, 2006. ACM Press.

[PO07] Fábio Policarpo et Manuel M. Oliveira : Relaxed cone stepping for relief mapping. *In* Hubert Nguyen, éditeur : *GPU Gems 3*, chapitre 18 - Relaxed Cone Stepping for Relief Mapping. Addison-Wesley, 2007.

[POC05] Fábio Policarpo, Manuel M. Oliveira et Joao L. D. Comba : Real-time relief mapping on arbitrary polygonal surfaces. *In SI3D '05 : Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 155–162, New York, NY, USA, 2005. ACM Press.

[Pom00] Alex Pomeranz : *ROAM using surface triangle clusters (RUSTiC)*. Thèse de doctorat, Department of Computer Science, University of California, Davis, juin 2000.

[PRPC09] Anna Paviotti, Filippo Ratti, Luca Poletto et Guido Maria Cortelazzo : Multispectral acquisition of large-sized pictorial surfaces. *Journal on Image and Video Processing*, 2009:1–1, 2009.

[QQZ+03] Huamin Qu, Feng Qiu, Nan Zhang, Arie Kaufman et Ming Wan : Ray tracing height fields. *In Computer Graphics International*, pages 202–209. IEEE Computer Society, 2003.

[RK09] Roland Ruiters et Reinhard Klein : Heightfield and spatially varying BRDF reconstruction for materials with interreflections. *Computer Graphics Forum*, 28(2):513–522, avril 2009.

[RLIB99] Martin Reddy, Yvan Leclerc, Lee Iverson et Nat Bletter : TerraVision II : Visualizing massive terrain databases in VRML. *IEEE Computer Graphics and Applications*, 19:30–38, 1999.

[RTG97] Holly Rushmeier, Gabriel Taubin et André Guéziec : Applying shape from lighting variation to bump map capture. *In* Julie Dorsey et Philipp Slusallek, éditeurs : *Eurographics Rendering Workshop 1997*, pages 35–44, New York City, NY, juin 1997. Eurographics, Springer Wien.

[Sam84] Hanan Samet : The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):187–260, 1984.

[Sam90] Hanan Samet : *The design and analysis of spatial data structures*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[SC00] Peter-Pike J. Sloan et Michael F. Cohen : Interactive horizon mapping. *In Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 281–286, London, UK, 2000. Springer-Verlag.

[SCD+06] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein et Richard Szeliski : A comparison and evaluation of multi-view stereo reconstruction algorithms. *In Computer Vision and Pattern Recognition*, pages 519–528. IEEE Computer Society, 2006.

[Sch98] Gernot Schaufler : Per-object image warping with layered impostors. *In* George Drettakis et Nelson Max, éditeurs : *Rendering Techniques '98*, Eurographics, pages 145–156. Springer-Verlag Wien New York, 1998.

[SG06] M. F. A. Schroders et R. v. Gulik : Quadtree relief mapping. *In GH '06 : Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, pages 61–66, New York, NY, USA, 2006. ACM.

[SKU08] László Szirmay-Kalos et Tamás Umenhoffer : Displacement mapping on the GPU — state of the art. *Computer Graphics Forum*, 27, 2008.

[SS92] H. Samet et R. Sivan : Algorithms for constructing quadtree surface maps. *In Proceedings 5th International Symposium on Spatial Data Handling*, pages 361–370, Charleston, août 1992.

[ST90] Peter Shirley et Allan Tuchman : A polygonal approximation to direct scalar volume rendering. *SIGGRAPH Comput. Graph.*, 24(5):63–70, 1990.

[SW06] Jens Schneider et Rüdiger Westermann : Gpu-friendly high-quality terrain rendering. *In Journal of WSCG*, volume 14, pages 49–56, 2006.

[Tat06a] Natalya Tatarchuk : Dynamic parallax occlusion mapping with approximate soft shadows. *In SI3D '06 : Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 63–69, New York, NY, USA, 2006. ACM Press.

[Tat06b] Natalya Tatarchuk : Practical parallax occlusion mapping with approximate soft shadows for detailed surface rendering. *In SIGGRAPH '06 : ACM SIGGRAPH 2006 Courses*, pages 81–112, New York, NY, USA, 2006. ACM.

[TIS08] Art Tevs, Ivo Ihrke et Hans-Peter Seidel : Maximum mipmaps for fast, accurate, and scalable dynamic height field rendering. *In SI3D '08 : Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 183–190, New York, NY, USA, 2008. ACM.

[TMJ98] Christopher C. Tanner, Christopher J. Migdal et Michael T. Jones : The clipmap : a virtual mipmap. *In SIGGRAPH '98 : Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 151–158, New York, NY, USA, 1998. ACM.

[TTK04] S. Tominaga, N. Tanaka et T. Komada : Imaging and rendering of oil paintings using a multi-band camera. *In Southwest Symposium on Image Analysis and Interpretation*, pages 6–10, 2004.

[Ulr00] Tatcher Ulrich : Rendering massive terrains using chunked level of detail control. *In Super-size it ! Scaling up to Massive Virtual Worlds (ACM SIGGRAPH tutorial Notes)*, New York, NY, USA, 2000. ACM Press.

[Wag03] Daniel Wagner : Terrain geomorphing in the vertex shader. *In ShaderX-2*. Wordware Publishing, 2003.

[Wel04] Terry Welsh : Parallax mapping with offset limiting : a per-pixel approximation of uneven surfaces. Rapport technique, Infiscape Corporation, 2004.

[WH03] Chris Wyman et Charles Hansen : Penumbra maps : approximate soft shadows in real-time. *In Proceedings of the 14th Eurographics workshop on Rendering*, pages 202–207, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[Wil83] Lance Williams : Pyramidal parametrics. *SIGGRAPH Computer Graphics*, 17(3):1–11, 1983.

[WLL+09] Tim Weyrich, Jason Lawrence, Hendrik P. A. Lensch, Szymon Rusinkiewicz et Todd Zickler : Principles of appearance acquisition and representation. *Foundations and Trends® in Computer Graphics and Vision*, 4(2):75–191, 2009.

[WMD+04] Roland Wahl, Manuel Massing, Patrick Degener, Michael Guthe et Reinhard Klein : Scalable compression and rendering of textured terrain data. *In Journal of WSCG*, pages 521–528, 2004.

[WWT+03] Lifeng Wang, Xi Wang, Xin Tong, Stephen Lin, Shimin Hu, Baining Guo et Heung-Yeung Shum : View-dependent displacement mapping. *In SIGGRAPH '03 : ACM SIGGRAPH 2003 Papers*, pages 334–339, New York, NY, USA, 2003. ACM.

[YJ04] Keith Yerex et Martin Jagersand : Displacement mapping with ray-casting in hardware. *In SIGGRAPH '04 : ACM SIGGRAPH 2004 Sketches*, page 149, New York, NY, USA, 2004. ACM.

Résumé

Les données à deux dimensions et demie ont pris une importance forte dans la représentation de certains types d'objets. Elles sont ainsi fréquemment employées pour modéliser des données topographiques ou scientifiques. Elles sont également utilisées dans les dispositifs d'acquisition à trois dimensions pour représenter les objets numérisés. Ces données présentent cependant un certain nombre de difficultés dans leur manipulation, et plus particulièrement pour leur visualisation.

Au cours de cette thèse, nous nous sommes attachés à développer des méthodes simples, mais efficaces, permettant la visualisation de ce type de données et plus spécialement celles issues de la numérisation de tableaux d'art. En plus de cette méthode de visualisation, nous avons développé un protocole complet permettant la numérisation de telles pièces, ainsi que le traitements des données obtenues, en vue de leur visualisation.

Afin de généraliser la méthode développée, nous avons également proposé une technique permettant la visualisation de données topographiques combinant un processus classique de rastérisation avec un rendu par lancer de rayons.

Les méthodes développées reposent ainsi toutes les deux sur un mécanisme adaptatif permettant de combiner différents algorithmes de rendu, afin d'optimiser les performances de visualisation. L'utilisation de ce mécanisme permet également une exploitation quasi-directe des données ce qui simplifie grandement leur utilisation.

Abstract

Heightfield data is now a common representation for several kind of virtual objects. Indeed, they are frequently used to represent topographical or scientific data. They are also used by 3-dimensional digitisation devices to store real objects. However, some issues are introduced by this kind of data during their manipulation and especially their visualisation.

During this thesis, we develop simple yet efficient methods to render heightfield data, especially data from art painting digitisation. In addition to the visualisation method, we propose a complete pipeline to acquire art paintings and to process data for the visualisation process.

To generalize the proposed approach, another rendering method is described to display topographical data by combining a rasterization process with a ray-casting rendering.

Both of the rendering techniques are based on an adaptive mecanism which combines several rendering algorithms to enhance visualisation performances. These mechanisms are designed to avoid pre-processing steps of the data and to make use of straightforward rendering methods.