Université de Montréal

Modélisation interactive par points d'objets complexes à partir d'images

par

François Duranleau

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la faculté des études supérieures

en vue de l'obtention du grade de

Maître ès sciences (M.Sc.)

en informatique

août 2002

Université de Montréal

Faculté des études supérieures

Ce mémoire de maîtrise intitulé

Modélisation interactive par points d'objets complexes à partir d'images

présenté par

François Duranleau

a été évalué par un jury composé des personnes suivantes :

Président:	Victor Ostromoukhov
Directeur de recherche:	Pierre Poulin
Membre:	Sébastien Roy

Mémoire accepté le 18 octobre 2002

Sommaire

Plusieurs techniques ont été développées pour afficher interactivement des objets graphiques très complexes (forêts, montagnes, villes, *etc.*) ou des objets plus simples mais très précis (personnes, sculptures, vases, *etc.*) grâce à l'utilisation de numériseurs tridimensionnels. Une de ces techniques d'affichage utilise des points pour échantillonner les objets eux-mêmes. Nous basant sur cette idée, nous présentons un système automatique et interactif permettant de reconstruire des modèles géométriques complexes. Nous employons une méthode stochastique de reconstruction tridimensionnelle par points tridimensionnels à partir d'un nombre restreint d'images sans contrainte sur la pose. Les résultats obtenus montrent que ce choix de représentation et que l'aspect stochastique de notre méthode de reconstruction sont efficaces pour cerner la complexité des détails d'objets réels.

Un aspect important de notre système est son interactivité. La complexité des objets que nous tentons de modéliser est une difficulté majeure pour l'ensemble des méthodes de reconstruction tridimensionnelle se basant sur la mise en correspondance de caractéristiques dans les images. Dans notre système, il est possible pour l'usager de guider le processus de reconstruction. Nous avons observé que ceci permet de l'accélérer et d'améliorer le résultat final.

Mots clefs :

représentation par points, reconstruction tridimensionnelle, modélisation à partir de photographies, raffinements automatiques et interactifs, rendu par points, matériel graphique.

Abstract

Many techniques have been developed to interactively display highly complex graphical objects (forests, mountains, cities, *etc.*) or simpler objects but with high precision (people, sculptures, vases, *etc.*) by using 3D scanners. One of these display techniques uses points to sample the objects themselves. Based on this idea, we present an automatic and interactive system which allows to reconstruct complex geometric models. We rely on a 3D point-based stochastic reconstruction method which uses a limited number of images without constraints on their pose. Our results show that a point-based representation and the stochastic aspect of our reconstruction method are efficient to grasp the complexity of real objects' details.

Another important aspect of our system is its interactivity. The complexity of objects that we attempt to model induces great difficulties for most 3D reconstruction methods based on feature correspondance in images. In our system, it is possible for the user to guide the reconstruction process. We have observed that it allows to speed up and to improve the final result.

Keywords :

Point-based representation, 3D reconstruction, photography-based modeling, automatic and interactive refinements, point-based rendering, graphics hardware.

Table des matières

Table des figures

Liste des tableaux

Remerciements

Je désire remercier de tout coeur mon directeur de recherche Pierre Poulin, de même que Marc Stamminger pour leurs travaux initiaux sur ce projet, leurs idées et leur support. Rien de ce projet n'aurait été possible sans eux. Je désire remercier Pierre aussi pour sa patiente, sa compréhension et sa constante bonne humeur. C'est un vrai plaisir que de travailler avec Pierre ! Je voudrais également remercier Chabane Tibiche et Marie-Claude Frasson pour leurs apports au projet.

Aux membres du laboratoire, merci pour votre agréable compagnie et pour nos partages de bonne musique ! Merci à mes parents pour leur support pendant ces deux années de travail. Merci en particulier à ma mère pour ses conseils grammaticaux.

Enfin, je désire remercier le FCAR pour son support financier.

Chapitre 1

Introduction

Depuis les débuts de l'infographie, le réalisme a toujours été l'une des grandes quêtes du domaine, qu'il s'agisse de réalisme des images de synthèse, des objets tridimensionnels, des simulations, *etc.* L'industrie du cinéma cherche constamment à produire des effets spéciaux plus convaincants les uns que les autres. L'industrie des jeux vidéo cherche à présenter aux joueurs des environnements toujours plus complexes. Les simulateurs de vol sont également très exigeants. Dans tous les cas, le besoin de représentations et d'affichage efficaces d'objets de plus en plus complexes est bien présent. Dans la dernière décennie, le réalisme des images de synthèse et des objets tridimensionnels s'est grandement amélioré, mais il est toujours très difficile et exigeant en travail de produire des objets complexes réalistes.

Par *objets complexes*, il est question d'objets qui comportent un grand nombre de détails de surface, des textures riches, des structures fines, de l'occultation, des trous, des discontinuités, *etc.* Les lits de fleurs, les arbres, des cordes nouées, les cheveux, une boule de papier, une pile de vêtements, *etc.*, sont tous des exemples d'objets que nous considérons comme très complexes dans le contexte de ce mémoire.

Traditionnellement, les objets tridimensionnels sont produits à l'aide de systèmes de modélisation, tels *Softimage|XSI* [Sof] et *Maya* de *Alias|Wavefront* [Ali]. Ils sont créés à l'aide de polygones, surfaces gauches, surfaces de subdivision ou de primitives géométriques (des coniques naturelles, par exemple). Bien qu'il soit possible de tout faire avec ceux-ci, les exigences au niveau de l'exploitation du temps de l'usager et de son talent sont très élevées. Malgré tout, la majorité des objets dans le domaine de l'infographie, jusqu'à présent, sont ainsi créés.

Plusieurs recherches ont été effectuées pour aider la production d'objets de diverses natures. Plusieurs d'entre elles exigent un équipement dispendieux ou beaucoup de mémoire, d'autres sont applicables seulement pour certaines classes d'objets, et enfin la plupart ne sont pas très bien appropriées pour les objets très complexes. Dans le cadre de ce mémoire, nous présentons une méthode qui se veut simple, flexible et accessible, *i.e.* une méthode simple d'usage et d'implantation avec laquelle nous pouvons produire un grand éventail de classes d'objets et qui ne requiert pas d'équipements dispendieux. Il s'agit d'une méthode de reconstruction tridimensionnelle par points tridimensionnels à partir d'un nombre restreint d'images sans contrainte sur la pose.

1.1 Structure du mémoire

Dans les prochains chapitres, nous abordons les motivations qui ont incité le développement de notre système tout en effectuant un bref survol des travaux antérieurs des domaines de la reconstruction tridimensionnelle et du rendu par points (chapitre 2). Puis, à l'aide d'un exemple d'une séance typique de modélisation avec notre système, nous donnons un aperçu global de notre méthode (chapitre 3). Les quatre chapitres suivants couvrent les détails de la méthode, soit l'initialisation (chapitre 4), la génération de points (chapitre 5), les techniques de raffinement (chapitre 6) et l'affichage (chapitre 7). Nous présentons ensuite des données statistiques et des exemples de résultats supplémentaires (chapitre 8). Finalement, nous effectuons un bilan des aspects positifs et négatifs de notre méthode lors de la conclusion (chapitre 9).

Chapitre 2

Motivations

Plusieurs méthodes ont été développées pour la génération d'objets complexes. Parmi celles-ci, nous retrouvons les systèmes-L pour les plantes [PL90, DHL+98]. Il s'agit d'un formalisme d'abord présenté par Lindenmayer [Lin68] similaire aux grammaires. Les systèmes-L sont très généraux et peuvent en théorie être employés pour modéliser n'importe quoi, leur puissance de calcul étant comparable à une machine de Turing. Cependant, ils sont plus appropriés pour modéliser les objets qui sont le produit d'un processus de croissance comme les plantes. Développer des modèles avec ce formalisme n'est cependant pas toujours très intuitif. Il s'agit de manipulations de bas niveau et il faut bien connaître le phénomène à reproduire.

D'autres méthodes ont été développées pour d'autres types de phénomènes, telle la modélisation de vagues [FR86], de terrains à l'aide de fractales [MKM89] et de villes [PM01] (les systèmes-L sont employés pour modéliser les rues et les édifices). Cependant, un inconvénient de ces méthodes est d'être restreintes à une classe d'objets et elles peuvent difficilement être adaptées pour d'autres classes.

Une alternative s'est présentée : le rendu à partir d'images dont, parmi les représentants classiques, nous retrouvons le *lightfield* [LH96] et le *lumigraph* [GGSC96]. Ces méthodes sont très générales et peuvent produire des images d'objets visuellement très complexes. Elles reparamétrisent l'information de radiance (couleur d'un pixel) à partir d'un grand nombre (des centaines) d'images pour afficher des nouvelles vues d'un objet sans nécessiter le support d'un modèle géométrique. Malheureusement, ces méthodes exigent beaucoup de mémoire et ont de la difficulté à traiter les cas d'occultation mutuelle et d'illumination avec d'autres objets. Ceci fait qu'elles sont difficiles à intégrer

dans un système de rendu typique. Une bonne partie de ces problèmes ont été récemment abordés par Matusik *et al.* [MPN+02], par contre l'exigence en mémoire est toujours bien présente. Ainsi il est question de 20 gigaoctets de données pour un seul objet [MPN+02].

Une autre avenue possible est la reconstruction tridimensionnelle. Plusieurs méthodes de reconstruction requièrent des équipements dispendieux ou imposent des contraintes sur l'acquisition des données. Néanmoins, les méthodes de reconstruction tridimensionnelle basées sur un nombre restreint (2 à 20) de photographies forment un moyen puissant pour modéliser des scènes réelles par la simplicité et la flexibilité de leurs méthodes d'acquisition. Avec la connaissance des paramètres des caméras, la correspondance entre les pixels des images est utilisée pour trouver la position tridimensionnelle d'un point et sa couleur de surface. Malgré le fait que de bons résultats ont été obtenus avec ces méthodes, l'extraction automatique d'objets complexes depuis un ensemble non contraint d'images demeure un problème difficile sans une solution complètement satisfaisante. L'intégration de l'usager dans le processus est un moyen de réduire plusieurs difficultés. Dans la section 2.1, nous effectuons un survol des méthodes de reconstruction tridimensionnelle.

Enfin, typiquement, les méthodes de reconstruction tridimensionnelle produisent soit un maillage ou un ensemble de voxels. Nous cherchons une représentation permettant de bien capturer la forme des objets complexes tout en l'affichant le plus efficacement possible. Notre choix s'est arrêté sur la primitive la plus simple qui soit : le point, que nous présentons à la section 2.2.

2.1 Reconstruction tridimensionnelle

Le domaine de la reconstruction tridimensionnelle est très vaste et beaucoup de travaux ont été effectués pour tenter de reconstituer en trois dimensions des objets réels. Nous effectuons dans cette section un très bref survol de l'état de l'art dans le domaine. Dans la section 2.1.1, nous abordons les procédés de numérisation tridimensionnelle, et la section 2.1.2 est consacrée à la présentation de l'apport de l'usager dans le processus de reconstruction.

Pour en savoir plus sur le domaine, des livres généraux devraient être consultés [Aya91, Fau93, TV98, KSK98], ou alors des revues de domaines plus spécifiques [ZTCS99].

Une collection importante d'articles est également maintenue par Price de USC (l'université de la Californie du Sud) [Pri02].

2.1.1 Numérisation tridimensionnelle

La numérisation tridimensionnelle regroupe les diverses méthodes d'acquisition de points tridimensionnels ou de surfaces à partir d'objets du monde réel. La figure 2.1 donne une taxonomie de l'ensemble de ces méthodes, inspirée de celle présentée par Dedieu [Ded01], à laquelle nous avons ajouté l'extraction de forme par photo-consistance. Dans les prochaines sections, nous effectuons un survol des quatre grandes catégories de numérisation, soit :

1. le palpage (moyen non optique par contact),
2. la télémétrie (moyen non optique sans contact),
3. la triangulation optique active (moyen optique actif),
4. les méthodes optiques passives (moyen optique passif).

Le palpage

Le palpage est une méthode non optique par contact. Ces méthodes font typiquement usage d'équipements tel un bras articulé (robotisé ou non) et calibré qui mesure les positions par mise en contact avec l'objet. Par bras calibré, il est question de connaître la position du capteur relativement à un point de référence, par exemple, sur sa base. Des exemples de tels équipements sont *Faro* [Far] et *MicroScribe* d'*Immersion* [Cor02]. Les méthodes de palpage ne conviennent cependant pas pour les formes fluides ou flexibles car le toucher entraîne une déformation. De plus, elles sont valides à l'intérieur d'une taille relative à celle du bras, *i.e.* la surface doit être atteignable par le bras. Enfin, ce sont des approches manuelles qui demandent passablement de travail.

La télémétrie

Les méthodes de télémétrie consistent à acquérir la forme d'objets par émission d'ondes et à mesurer le temps que l'émission prend pour revenir au capteur, en supposant que la vitesse de propagation des ondes soit connue et constante. Ce principe est employé pour les radars et les sonars. Les problèmes de ces méthodes sont reliés à l'absorption

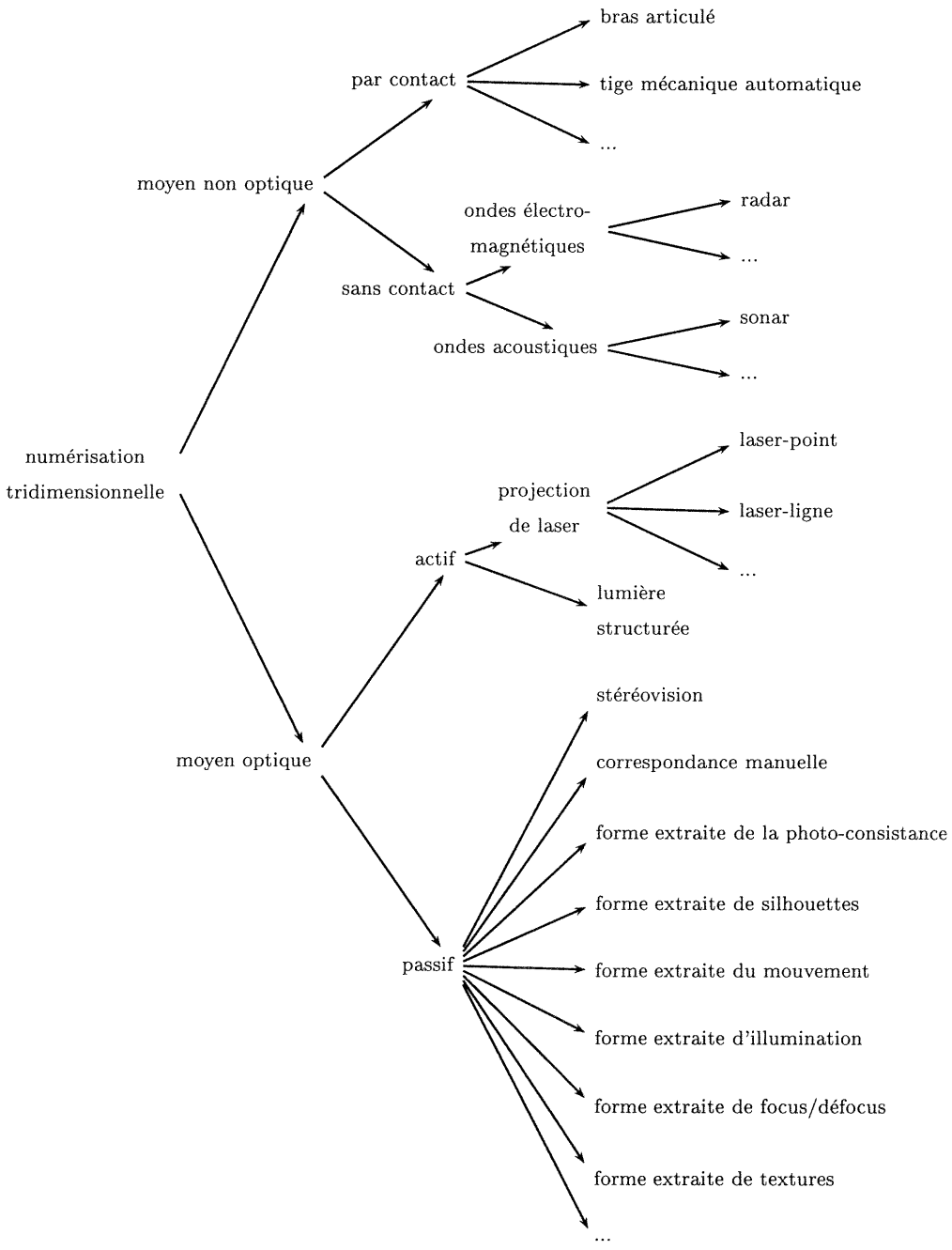


FIG. 2.1 – Taxonomie des méthodes de numérisation tridimensionnelle inspirée de Dedieu [Ded01].

des ondes ou au captage du retour (il suffit de penser aux avions militaires furtifs tel le fameux F-117). Les modèles ainsi reconstruits sont généralement moins précis et ces techniques éprouvent des problèmes avec des structures fines.

La triangulation optique active

Les méthodes de triangulation optique active procèdent par l'intermédiaire d'une source lumineuse et d'une caméra observant la scène. La source lumineuse émet typiquement un faisceau concentré de lumière tel un laser ou encore une lumière structurée, *i.e.* une illumination de l'objet par des motifs connus de lumière. Dans tous les cas, nous connaissons les positions relatives de l'émetteur et de la caméra. Sachant la direction d'un point observé (la reconnaissance du point dépend de la méthode d'émission lumineuse) pour deux positions données, soit la caméra et l'émetteur, nous pouvons retrouver sa position tridimensionnelle en intersectant les deux vecteurs.

Parmi les équipements pour effectuer cette numérisation, nous retrouvons :

- les lasers-point, tel *VIVID* 700 et 900 de *Minolta* [Min02] ;
- les lasers-ligne, tel *Whole body scanner* de *Cyberware* [Cyb] ou *FastSCAN* [Pol] ; la ligne est obtenue par l'intersection de l'objet avec un plan tridimensionnel lumineux, et la triangulation est effectuée pour chaque "point" de la ligne ;
- les projecteurs à lumière structurée, tel *Minolta 1500 flash 3D* [Min02] ; le motif, tel un code-barre avec sigles codés, permet d'identifier des lignes similairement aux lasers-ligne, seulement ici il n'est pas nécessaire de balayer l'objet avec le motif. Une méthode d'acquisition interactive [RHHL02] a été présentée récemment.

L'ensemble de ces appareils permettent de produire une quantité impressionnante de points de manière précise. Cependant, un travail de post-traitement est nécessaire pour pouvoir fusionner les numérisations multiples, nécessairement partielles, réalisées sous divers angles de l'objet à reconstruire. De plus, il faut souvent simplifier ou éliminer les objets ou les détails non désirés ou parasites. Un autre post-traitement est requis pour générer des maillages (*meshes*), quoique ce n'est pas nécessaire pour certaines applications, tel le rendu par points (section 2.2). Enfin, la réalisation du positionnement automatique (*fitting*) de primitives (cônes, cylindres, boîtes, *etc.*) peut être requise.

Les méthodes optiques passives

Ces méthodes reposent sur le traitement de multiples images ou d'une séquence vidéo où le point de vue change ou alors où il y a déplacement d'objets. Au niveau de l'utilisation, ceci offre plus de flexibilité que les autres méthodes de numérisation tridimensionnelle, notamment par la capacité de reconstruire un objet avec un équipement peu dispendieux et dont les prises de vue peuvent être *a priori* arbitraires, quoique certaines méthodes imposent des contraintes.

La méthode classique de triangulation passive est la stéréovision [LH81]. À partir de vues légèrement décalées et généralement précisément calibrées (*i.e.* nous connaissons leur position relative les unes par rapport aux autres), l'approche consiste à mettre automatiquement en correspondance des pixels (par corrélation) et à calculer la disparité (information sur la profondeur) de chacun d'eux par triangulation. Plusieurs problèmes peuvent se poser pour la bonne mise en correspondance des pixels causés soit par le bruit dans les images, les occultations, les propriétés de réflectance des surfaces, *etc.* Ces problèmes ne sont pas exclusifs à la stéréovision, mais à toute méthode se basant sur des mises en correspondance de pixels. Il est possible de faire la mise en correspondance manuellement : l'usager spécifie des points saillants dans les images et donne leur disparité ou alors leur véritable position tridimensionnelle. Ceci évite l'ambiguïté de l'approche automatique, mais demande du temps de la part de l'usager pour effectuer la mise en correspondance.

Un autre problème relié à la stéréovision est la restriction imposée sur la proximité des positions des caméras, et par le fait même nous pouvons reconstruire seulement une "face" de l'objet (visible dans les deux images). Ce problème est également présent dans la triangulation optique active, et ainsi un post-traitement supplémentaire est requis pour aligner et fusionner les reconstructions des diverses "faces" de l'objet. De plus, un lissage est souvent effectué pour améliorer la qualité de la reconstruction. Ceci est bon pour les surfaces lisses, mais néfaste pour les objets plus complexes, tel un buisson.

Les méthodes d'extraction de forme par photo-consistance [SD99, KS00, Dye01, SCMS01] sont un autre type d'approches de grand intérêt pour nous. Parmi ces méthodes, la coloration de voxels (*voxel colorization*) [SD99] offre un moyen simple de reconstruction *photo-consistante* moyennant une contrainte sur les prises de vue : la scène à reconstruire doit se trouver entièrement en dehors de l'enveloppe convexe formée

par les points de vue des caméras. Une scène sera dite *photo-consistante* si, pour chaque image et pour chaque pixel dans lequel la scène projette, la couleur de ce dernier est la même (ou presque) que la couleur projetée par la scène reconstruite. La méthode procède en "colorant" chaque voxel d'un volume prédéterminé qui englobe la scène. Les voxels sont traités par couches (un plan) successives à partir des voxels qui sont les plus près des points de vue afin de traiter convenablement la visibilité. Un voxel sera accepté (coloré) si ses projections dans chaque image dans laquelle il est visible sont consistantes, *i.e.* de même couleur (à un epsilon près). Un voxel est visible dans une image si aucun autre voxel n'a déjà été projeté dans les pixels où il projette. Si les projections ne sont pas consistantes, ce voxel est exclu, *i.e.* il est vide, il ne fait pas partie de la scène.

La sculpture d'espace (*space carving*) [KS00] généralise cette méthode sans aucune contrainte sur les points de vue. Elle procède essentiellement comme la coloration de voxels, mais le plan de génération n'est pas choisi en fonction des points de vue. Ce sont plutôt ces derniers qui sont ajoutés à une liste de points de vue candidats au fur et à mesure que le plan se déplace. Plus d'une passe (six) est nécessaire pour traiter convenablement toutes les directions. Chaque passe est effectuée selon les trois axes principaux du volume englobant la scène. À chaque nouvelle passe, les voxels acceptés des passes précédentes se trouvant sur le plan actuel doivent être revalidés. Cette méthode offre plusieurs des propriétés désirées dans notre contexte : des images provenant de simples caméras, points de vue arbitraires, et capacité de reconstruire des objets complexes. Cependant, cette méthode peut s'avérer exigeante en mémoire et sa grille régulière de taille fixe peut nuire à la bonne capture des détails fins.

Les méthodes basées uniquement sur la photo-consistance ne peuvent produire mieux comme solution que l'*enveloppe photographique* (*photo hull*). Il s'agit de l'enveloppe minimale qui englobe toutes les scènes photo-consistantes possibles à l'intérieur du volume prédéterminé qui englobe la scène véritable. Ces méthodes sont dites "à implication minimale". Moins il y a d'images disponibles, pire en sera la qualité de la reconstruction.

D'autres méthodes exploitent des informations supplémentaires pour soit aider ces méthodes soit s'en servir pour développer de nouvelles méthodes. Parmi celles-ci, il y a :

- **forme extraite de silhouettes :** Connaissant la silhouette des objets dans chaque image, nous pouvons en estimer la forme [CB92, Sze93, Zhe94, SP98,

SVZ00]. Il peut être difficile d'extraire certaines concavités des objets avec ces méthodes.

- **forme extraite du mouvement :** À l'aide d'une caméra vidéo qui produit une suite d'images où nous localisons des points, des segments, *etc.*, il faut ensuite calculer automatiquement le parcours de la caméra (*tracking*) ainsi que les coordonnées tridimensionnelles des primitives identifiées sur la séquence d'images [TK92, TF87].
- **forme extraite d'illumination :** Il est souvent possible de calculer par approximation l'illumination d'une surface par un modèle mathématique simple, de même que pour les conditions d'éclairage et de visualisation. Ainsi, à partir d'une photographie, l'approche consiste à retrouver la géométrie de l'objet. La difficulté est l'établissement cohérent du modèle géométrique avec l'illumination perçue sur la photographie [HB89].
- **forme extraite de focus/défocus :** Les objets au foyer sont plus nets que les autres et donc, connaissant les propriétés optiques de l'objetif de la caméra et en prenant plusieurs photographies en ne changeant que le focus, il est possible d'extraire de l'information sur la profondeur des objets [EL93].
- **forme extraite de textures :** S'aidant de données statistiques (régularité, symétrie, périodicité), il est possible d'inférer l'orientation d'échantillons de surface [KC89].

2.1.2 Intégration de l'utilisateur au processus de reconstruction

Du fait que la majorité des applications de la reconstruction tridimensionnelle à partir d'images sont dans le domaine de la reconnaissance d'objets ou l'évitement de collisions en robotique, la plupart des techniques de numérisation tridimensionnelle ont été développées pour être complètement automatiques. De plus, des résultats partiels, et donc non photoréalistes, sont souvent satisfaisants pour ces applications. Le photoréalisme des surfaces extraites n'a reçu plus d'attention que depuis la dernière décennie.

L'intégration de l'usager au processus de reconstruction peut souvent aider à satisfaire les exigences additionnelles du photoréalisme. Il y a toujours des risques d'ambiguïté de mises en correspondance avec lesquelles les méthodes automatiques ont de la

difficulté. Ces dernières requièrent des heuristiques pour réduire les ambiguïtés. L'usager peut alors aider le système en apportant des mises en correspondance manuelles. Ses apports ne sont pas limités qu'à l'aide à la mise en correspondance. Il peut aussi exploiter ses connaissances sur la nature de la scène pour apporter un jugement qualificatif et guider le processus de reconstruction.

Un certain nombre de systèmes dans le domaine académique [DTM96, SWI97, POF98, GMMB00, SRDT01, Ded01] et de systèmes commerciaux [Can, Phob, Rea] ont été développés avec l'idée d'intégrer l'usager au processus de reconstruction. Cependant, à l'exception du raffinement des modèles basés sur la géométrie épipolaire de Façade [DTM96], seulement de simples surfaces polygonales sont extraites de ces systèmes. Le positionnement manuel de polygones exige clairement beaucoup de travail et c'est souvent une tâche impossible pour les objets complexes, tels des arbres. La complexité se retrouve en fait sous forme de textures. Ceci est valide surtout pour les scènes architecturales, *i.e.* polygonales.

2.2 Représentation par points

L'idée d'employer le point comme primitive pour l'affichage plutôt que des polygones ou d'autres primitives plus complexes a d'abord été soulevée par Levoy et Whitted [LW85]. Cette primitive avait déjà été employée pour le rendu de phénomènes précis, tels la fumée [CHP+79], les nuages [Bli82], le feu [Ree83] et les arbres [Smi84], mais ici il était question de l'employer de façon générale pour tout type d'objets. L'une de leurs motivations provient du fait que les courbes et les surfaces traditionnelles (polygones, splines) ne représentent pas aisément certains types d'objets naturels (par exemple le feu et les nuages). En effet, plus la complexité d'un objet augmente, moins il y a de cohérence à exploiter, ou alors le coût pour l'exploiter dépasse le temps ainsi gagné (par exemple, s'il y a plus d'un polygone par pixel lors de l'affichage). Leur approche se limitait cependant aux surfaces différentiables. Une autre solution à ce problème, le rendu à base d'images, a été préférée pour la représentation d'objets complexes.

Ce n'est que plusieurs années plus tard que de nouveaux développements ont été effectués dans cette direction. Grossman et Dally [GD98] cherchaient une alternative au rendu à base d'images, trop coûteux en mémoire, pour effectuer un rendu interactif sans matériel graphique avancé. Ils ont développé un algorithme basé sur la représentation

par points. La surface de l'objet est échantillonnée selon plusieurs projections orthographiques sous divers angles de vue. La densité des échantillons est ainsi à peu près uniforme de sorte que pour une résolution d'affichage cible, il y ait au moins un point qui se projette dans chaque pixel où l'objet serait en théorie projeté. Des algorithmes multi-résolution sont employés pour détecter et remplir les trous qui pourraient se produire dans l'image. Leurs idées ont été reprises par Pfister *et al.* [PZvBG00]. Il est question d'un système semblable mais plus avancé. Ils suggèrent un pipeline graphique adaptable pour une implantation matérielle et qui est en mesure de produire des images avec peu d'aliassage et de façon interactive sans accélération matérielle. L'affichage de surfaces transparentes y est cependant impossible. L'année suivante, Zwicker *et al.* [ZPvBG01] apportent plusieurs améliorations au niveau de l'anti-aliassage et de la transparence en présentant une étude rigoureuse sur les fonctions de texture (*texture functions*) à chaque point. Dans leur système de rendu, les trous et l'anti-aliassage sont réduits en effectuant un rééchantillonnage des fonctions de texture en espace image, après la projection des points. L'interactivité du rendu n'est plus tellement de la partie cependant, bien qu'il ne faille que quelques secondes pour l'effectuer.

Rusinkiewicz et Levoy [RL00] ont développé une autre approche se servant d'une hiérarchie de sphères englobantes. Leur motivation principale était l'affichage de modèles géométriques très complexes au niveau de détail de leur surface et généralement trop exigeants en mémoire [LPC+00]. Les numériseurs tridimensionnels modernes produisent une très grande quantité de points et il faut une méthode pour les afficher rapidement. Chaque point du modèle est regroupé avec ses voisins dans une sphère. Puis le même procédé est repris pour les sphères, regroupées dans des sphères plus grandes, et ainsi de suite jusqu'à ce qu'il n'y ait plus qu'une seule sphère. L'affichage est effectué progressivement par niveau de la hiérarchie, en débutant par le haut. Chaque sphère ou point est affiché avec un *splat*, *i.e.* un carré, un disque, ou toute primitive simple, dont la dimension dépend du niveau de la hiérarchie.

L'un des problèmes reliés à l'ensemble de ces méthodes est qu'elles se basent sur un échantillonnage fixe et précalculé de points pour un modèle géométrique donné, ce qui limite leur application à des scènes statiques. Stamminger et Drettakis [SD01] ont développé une méthode d'échantillonnage dynamique pour diverses classes d'objets. Cet échantillonnage permet de générer un nombre de points suffisant pour couvrir les pixels

à l'écran, mais sans générer de surplus en terme de densité de points. *A priori*, aucun point n'a besoin d'être généré d'avance. Il suffit d'avoir une méthode de paramétrisation bidimensionnelle de l'objet (elle peut être effectuée par morceaux) et une métrique qui permet d'évaluer la suffisance de la densité de l'échantillonnage. Cette métrique peut cependant être difficile à développer, si même elle existe, pour certaines classes d'objets.

Enfin, il y a aussi une tendence vers l'emploi de points non seulement pour des fins de rendu mais aussi pour représenter la forme des objets. Alexa *et al.* [ABCO+01] présentent les surfaces d'ensembles de points (*point set surfaces*). Ces ensembles de points sont également dynamiques et il est possible de produire des scènes à divers niveaux de résolution. Ils se servent de surfaces de régression de moindres carrés (*moving least square surfaces*) pour rééchantillonner la surface. Linsen [Lin01] présente une autre façon de simplifier ou lisser un ensemble de points avec des opérateurs plutôt liés au traitement de signal pour les maillages [Tau95, KCVS98, GSS99]. Il présente également quelques méthodes pour modifier interactivement les ensembles de points, les filtrer, ou appliquer des opérations CSG (*Constructive Solid Geometry*) [FvDFH96]. Au niveau du traitement des points, Pauly et Gross [PG01] présentent une approche spectrale pour le faire. Ils subdivisent la surface en pièces qu'ils paramétrisent en deux dimensions. De là, ils font leur traitement par transformée de Fourier. Leurs idées ont été intégrées par Zwicker *et al.* [ZPKG02] dans un système d'édition de surfaces représentées par des points. C'est un système interactif qui repose sur la paramétrisation locale de la surface et sur le rééchantillonnage dynamique pour appliquer diverses transformations sur les points (couleurs, légers déplacements, filtrage, *etc.*).

Dans tous ces derniers développements, la motivation majeure est que l'emploi d'une représentation par points évite de calculer un maillage. Comme les numériseurs tridimensionnels produisent beaucoup de points, ceci réduit une importante étape de calcul et nous n'avons pas besoin de structures très complexes pour emmagasiner les points. Bien qu'avec ceux-ci nous perdions l'information d'adjacence, tous ces développements illustrent que nous pouvons nous en tirer autrement. De plus, la représentation par points ne se limite pas aux surfaces. Elle se prête également bien pour les modèles volumétriques. Il s'agit d'une très bonne représentation pour les scènes complexes avec beaucoup de discontinuités et de petits détails, telles les scènes naturelles.

Ainsi, pour notre système et pour toutes ces raisons, nous avons opté pour le point

comme primitive de base. Nous évitons la régularité d'échantillonnage spatial des voxels et la complexité de manipulations de maillage, tout en permettant un rendu de qualité.

Chapitre 3

Séance de modélisation

Afin de donner un aperçu de notre système, nous présentons dans ce chapitre un exemple d'une séance typique de modélisation. Nous présentons d'abord les initialisations requises à la section 3.1 puis les diverses étapes effectuées dans notre exemple sont présentées à la section 3.2.

3.1 Initialisation

L'usager prend une série de photographies d'un objet qu'il désire reconstruire. La scène servant d'exemple ici est un monceau de cordes d'escalade. Quatorze photographies ont été prises. La figure 3.1 montre quatre de ces images. Les images sont ensuite calibrées avec *ImageModeler* de *RealViz* [Rea] pour déterminer les paramètres de caméra correspondant à chaque image dans un système de coordonnées tridimensionnel commun. Dans ce système, l'usager doit définir une zone d'intérêt (volume de reconstruction, un cylindre ici) où reconstruire la scène, illustré en fil de fer et avec sa boîte englobante à la figure 3.2 (a). Les détails concernant l'initialisation du système sont présentés au chapitre 4.

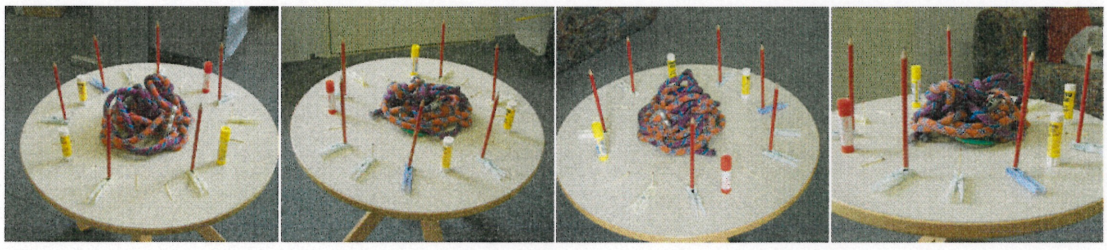
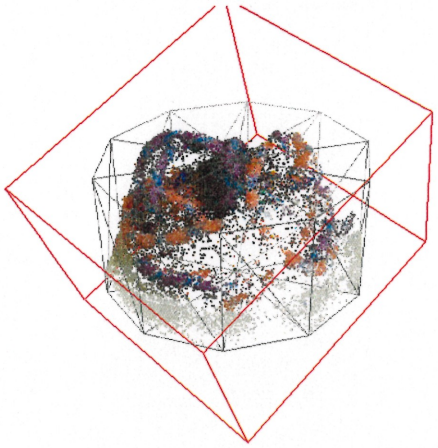


FIG. 3.1 – Quatre des 14 images utilisées pour l'exemple des figures 3.2 et 3.3.

3.2 Génération et raffinement

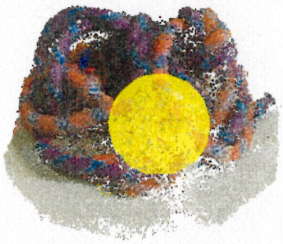
La reconstruction de la scène s'effectue typiquement avec une alternance de génération de points et de raffinement des points générés. Les figures 3.2, 3.3 et 3.4 illustrent un cheminement typique. Les statistiques (nombre de points, seuils, temps d'exécution, *etc.*) sont données au chapitre 8 (section 8.1.1).

La génération de points est effectuée en prenant une position aléatoire tridimensionnelle dans le volume de reconstruction et en vérifiant sa consistance avec l'ensemble des images dans lesquelles le point est *visible*, de façon similaire à la sculpture d'es-



(a) Ensemble initial de 26 638 points générés avec la méthode de base (chapitre 5). Le volume de reconstruction (section 4.3) avec sa boîte englobante sont affichés en fil de fer pour illustrer la zone de reconstruction.

(b) Poursuite de la génération en ajoutant au processus une génération par perturbation (section 6.1.1). Il y a maintenant 103 809 points.



(c) L'usager identifie une région jusqu'ici négligée (section 6.2.1).

(d) Génération additionnelle de points dans la région d'intérêt, qui a été déplacée à plusieurs endroits.

FIG. 3.2 – Séance typique de modélisation.

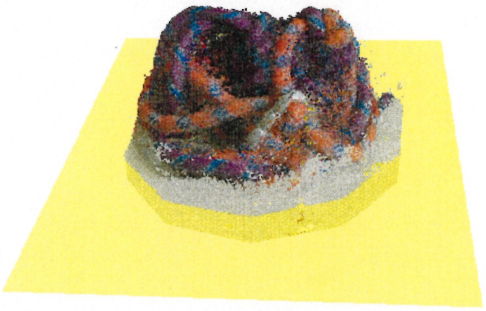
(e) Fusion des points (section 6.1.2) pour réduire les redondances. 13% des points ont été éliminés avec une grille de 293 par 224 par 256 voxels.

(f) Un zoom est effectué pour voir le modèle à un niveau de détail plus fin. Les trous entre les points deviennent plus apparents.

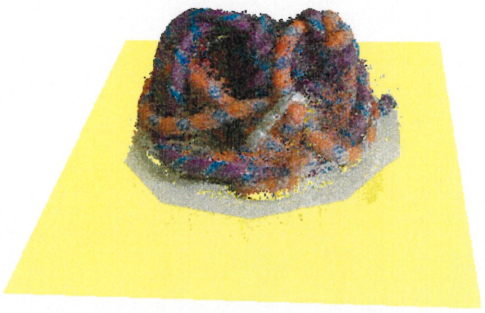
(g) Remplissage de la vue du zoom de la figure précédente en (f) (section 6.2.2).

(h) De retour à la vue initiale, plus de points sont générés avec la méthode de base ou par perturbation, surtout à l'aide de la région d'intérêt, et quelques fusions sont effectuées.

FIG. 3.3 – Séance typique de modélisation (suite).



(i) Positionnement d'un plan qui servira à corriger la position des points représentant la table (section 6.2.4)



(j) Projection sur le plan d'un ensemble de points sélectionnés par l'usager.

(k) Génération de points sur le plan pour enrichir la surface de la table.

(l) Vers la fin de séance, après une fusion et un nettoyage manuel de points à des positions aberrantes (section 6.2.1).

FIG. 3.4 – Séance typique de modélisation (suite).

pace [KS00]. Ainsi, il faut que les couleurs des pixels dans lesquels le point est projeté soient similaires pour un nombre suffisant d'images. La visibilité est déterminée à l'aide de cartes de profondeurs générées pour chaque image à partir de l'ensemble actuel de points tridimensionnels. Ces cartes sont calculées sur demande ou après un certain nombre de points testés. Enfin, pour compléter le processus de génération, un *nettoyage* est effectué de temps à autres, typiquement après la génération des cartes de profondeurs, afin d'éliminer les points qui ne sont plus visibles. Les points de la figure 3.2 (a) ont été générés ainsi. Les détails de la génération de points sont présentés au chapitre 5.

L'usager peut observer l'ensemble de points interactivement en trois dimensions avec possiblement une superposition avec les images d'entrée (chapitre 7). L'usager peut se servir de ses observations du processus de reconstruction de la scène pour le guider. À la figure 3.2 (b), l'usager a effectué un raffinement de la méthode de génération : les positions actuelles des points servent comme germes pour générer des points dans leur voisinage (génération par perturbation, section 6.1.1). À la figure 3.2 (c), l'usager identifie une région à l'aide d'une sphère, région jusqu'ici négligée par le processus aléatoire de génération. En générant des points seulement à l'intérieur de cette sphère (section 6.2.1) et en la déplaçant à une quinzaine d'endroits désirés, il obtient ainsi plus rapidement un meilleur résultat pour ces régions (à la figure 3.2 (d)).

Alors que certaines régions manquent de points à cause de visibilités complexes, structures fines, spécularités, *etc.*, d'autres au contraire génèrent rapidement une plus forte densité de points. Afin de réduire les redondances parmi les points générés, l'usager peut effectuer une fusion des points (section 6.1.2). Cette fusion s'effectue par l'intermédiaire d'une grille régulière où, à l'intérieur de chaque voxel, nous conservons le point le plus consistant. La figure 3.3 (e) montre le résultat d'une telle fusion. Il y a peu de différences visuellement avec le résultat de l'étape précédente 3.2 (d) (le contraire n'est généralement pas souhaitable).

L'usager peut effectuer un zoom sur la scène pour évaluer le résultat à un niveau de détail plus fin. À la figure 3.3 (f), le zoom révèle plusieurs trous entre les points, ce qui est naturel avec cette représentation. Il est possible de les combler plus rapidement avec une méthode de remplissage pour la vue courante (section 6.2.2). Ici, des points sont générés et validés dans les pixels vides. À la figure 3.3 (g), le résultat de l'application

de cette méthode est présenté.

Dans notre exemple, à la figure 3.3 (h), l'usager a poursuivi sa séance de modélisation à l'aide de diverses méthodes de génération (principalement à l'aide de la région d'intérêt). Lorsque le résultat est satisfaisant, l'usager peut vouloir corriger les positions erronées de certains points causées par des ambiguïtés sur les mises en correspondance (section 6.2.4). Un exemple typique de ces ambiguïtés sont les points issus de la surface de la table sur laquelle repose les cordes. À la figure 3.4 (i), l'usager positionne un plan qui servira à corriger les points que l'usager a sélectionnés, tel que présenté à la figure 3.4 (j). La sélection est effectuée de façon similaire à la spécification de la région d'intérêt de la figure 3.2 (c). Ensuite, à la figure 3.4 (k), il peut se servir de ce même plan pour générer d'autres points sur sa surface afin de combler les détails de la surface de la table.

Finalement, à la figure 3.4 (l), l'usager effectue un nettoyage manuel des points en retirant ceux à des positions jugées aberrantes (section 6.2.1). Une dernière fusion a également été effectuée pour produire le résultat final.

Nous n'avons pas présenté dans cet exemple la totalité des techniques de raffinement de notre système disponibles à l'usager. Le chapitre 6 est consacré à l'ensemble de ces techniques.

Chapitre 4

Initialisation

Les données d'entrée de notre système sont un ensemble d'images numériques calibrées ainsi que la définition d'un volume qui englobe la scène à reconstruire. Nous décrivons dans ce chapitre la prise de ces images (section 4.1), leur calibration (section 4.2) et la définition du volume de reconstruction (section 4.3).

4.1 Acquisition

Les données primaires de notre système sont un ensemble d'images numériques d'une même scène. Nous avons besoin d'images de scènes statiques (où rien ne bouge) et où il est important que pour un même objet d'une scène, la couleur de pixels correspondants soit identique (idéalement) ou semblable d'une image à l'autre. Cependant, plusieurs difficultés se posent pour atteindre cette invariance :

- **la réflectance des objets :** tout n'est pas que diffus, et la couleur d'un objet peut dépendre des angles sous lesquels on le regarde par rapport aux angles d'incidence de l'éclairage ;
- **l'occultation :** un objet peut être visible dans un sous-ensemble des images seulement ;
- **la discrétisation des images en pixels :** chaque image correspond à un nombre fini de pixels, et donc chacun d'eux correspond à une intégration des couleurs qui sont projetées dans leur espace, ce qui donne lieu à des variations, en particulier dans les régions riches en texture ou à la frontière des objets dans l'image ;
- **la réflexion et la réfraction :** les objets miroirs ou semi-transparents posent beaucoup de problèmes car ce que nous voyons à leur surface dépend du point de vue.

Malgré tout, certains résultats pour reconstruire la scène sont souvent possibles, d'où le développement de notre système.

Ces images peuvent provenir de diverses sources : photographies (numériques ou numérisées), images de synthèse, dessins ou peintures (de précision), tomographies (*CT scans*, *SPECT*), ou tout système d'imagerie. Les prochaines sous-sections donnent plus de détails sur quelques-unes de ces méthodes d'acquisition. Bien entendu, plus nous avons d'images, plus nous devrions obtenir une reconstruction de meilleure qualité, mais ceci implique une plus grande consommation de ressources en mémoire, temps d'exécution (section 5.2), et temps de calibration (section 4.2).

4.1.1 Photographies

Pour des scènes réelles (*i.e.* notre but premier), la méthode de prédilection actuelle est la caméra numérique. Il est toujours possible d'employer une caméra classique, mais cela requiert une étape additionnelle pour numériser les photographies ainsi obtenues. Nous pouvons employer un numériseur, mais nécessairement la qualité finale des images produites en souffre (c'est comme prendre une photographie d'une photographie). Cependant, il existe des technologies pour numériser directement le négatif d'une photographie, tel *Kodak PhotoCD* [Phoa]. Cela demeure une étape de numérisation, mais la qualité des résultats rivalise avec la photographie numérique, sans compter qu'il est possible d'atteindre des résolutions beaucoup plus grandes. Cependant, les frais de développement sont assez élevés, le délai de développement est en général plus long que le simple développement de films, et ce service n'est pas offert partout. La photographie numérique, quant à elle, ne coûte rien en développement, et elle offre beaucoup plus de flexibilité au niveau de la prise des photographies.

Dans les deux cas (photographies numériques ou numérisées), il faut s'assurer de préserver les couleurs de la scène à reconstruire d'une image à l'autre (section 5.2). Ceci implique que les conditions d'éclairage et le temps d'exposition (ouverture et vitesse d'obturation) doivent être les mêmes pour toute image.

Pour aider davantage l'invariance des couleurs, les photographies devraient être prises en même temps avec des appareils identiques. Ceci peut s'avérer coûteux en équipement et, en pratique, ce n'est pas d'une nécessité absolue, en particulier si les photographies sont prises dans un environnement où l'éclairage est contrôlé (par exemple, dans une pièce n'ayant aucune fenêtre sur l'extérieur). Pour les scènes extérieures, si

chaque photographie est prise dans un intervalle de temps raisonnable, l'éclairage ne devrait pas poser un problème (il faut tout de même faire attention à l'ombrage des nuages). Cependant, il faut faire attention à d'autres facteurs tel le vent, en particulier si nous prenons des images d'un arbre, par exemple.

4.1.2 Images de synthèse

Il est également possible d'employer des images de synthèse. Bien entendu, avec notre système, nous voulons en réalité extraire un modèle géométrique qui servira à produire de telles images, et donc l'usage d'images de synthèse en entrée peut paraître futile. Cependant, dans ces conditions, nous avons un contrôle absolu sur la précision et la qualité des images produites, et donc il est facile de produire des cas particuliers afin de tester nos algorithmes et établir des comparaisons entre une situation idéale (images de synthèse) et une situation courante (photographies). C'est donc dans cette optique que nous employons également des images de synthèse.

4.1.3 Caméras à diapason dynamique étendu

L'usage de caméras à diapason dynamique étendu (*high dynamic range*) [Ham01, Sph02, Hit00, GW00] devrait nous fournir des images de meilleure qualité en évitant la sur-saturation ou la sous-saturation des couleurs. À défaut d'avoir de tels appareils, il est possible de construire de telles images à diapason dynamique étendu à partir d'images prises à divers temps d'exposition [DM97, DFL02]. Cette dernière méthode produit des images de meilleure qualité, mais exige plus de travail.

4.2 Calibration

La calibration de caméras permet de trouver la correspondance entre un ensemble de points tridimensionnels et leur projection, et ensuite de déduire les paramètres des caméras. Nous distinguons souvent les paramètres intrinsèques et extrinsèques, car très souvent ces derniers sont inconnus alors que les premiers sont parfois donnés par le fabricant du modèle de caméra. Les paramètres intrinsèques correspondent aux caractéristiques optiques, géométriques et numériques (discrétisation en pixels) de la caméra, alors que les paramètres extrinsèques donnent la position et l'orientation de la caméra dans un système de coordonnées de référence. Après la calibration, nous obtenons une correspondance de tout point tridimensionnel de l'espace de référence

commun vers l'espace bidimensionnel de chaque image. Cette correspondance est typiquement exprimée sous la forme d'une matrice de projection.

Il existe une vaste littérature dans le domaine de la vision par ordinateur sur la calibration de caméras [TV98, KSK98, Fau93, Aya91]. Certaines techniques se basent sur des points saillants, d'autres sur la stéréo-vision, sur l'appariement global, *etc.* Pour les fins de notre système, n'importe quelle méthode de calibration peut êre utilisée, qu'elle soit automatique ou interactive, pourvu qu'elle soit robuste. Car, étant donné que notre système se base surtout sur la correspondance des couleurs entre les images, et donc d'une projection précise des points générés (section 5.2), il est important que la calibration soit la plus précise possible. Ce n'est pas suffisant cependant pour avoir une bonne reconstruction. Nous dépendons également du type de scène à reconstruire et de la qualité des images d'entrée (section 4.1).

Étant donné que typiquement les scènes cibles de notre système sont complexes, et donc difficiles pour les systèmes de calibration automatique, nous avons opté pour une approche plus interactive, en supposant que l'usager soit en mesure d'identifier facilement et de façon robuste un ensemble de points caractéristiques sur l'ensemble des images. Un point caractéristique correspond à un point sur un objet que nous retrouvons dans plusieurs images. De tels points se présentent plus rarement dans les scènes naturelles, et il faut alors ajouter des petits objets de calibration dans la scène qui doivent être visibles dans la majorité des images et à partir desquels il est plus facile de spécifier ces points caractéristiques.

Plusieurs techniques de calibration de caméras ont été présentées dans le domaine de l'infographie [DTM96, GGSC96, Ded01]. En pratique, nous avons employé le logiciel *ImageModeler* de *RealViz* [Rea], qui s'est avéré suffisamment robuste pour nos scènes. L'usager doit spécifier en moyenne 10 à 12 marqueurs (points caractéristiques ou saillants). Le tout exige cependant un certain temps de travail pour calibrer une scène, soit une séance d'environ une vingtaine de minutes pour des scènes où il y a des marqueurs facilement identifiables ou pour un petit nombre d'images (moins de 16), et jusqu'à une heure pour des scènes plus complexes ou avec un plus grand nombre d'images.

Il est à noter que cette étape n'est généralement pas nécessaire pour les images de synthèse car il est rare que nous ne puissions pas extraire tous les paramètres de

caméra (matrice de projection) du logiciel qui a servi à produire les images. Le modèle de caméra employé dans ces cas-ci est généralement simplifié par un modèle de caméra de type sténopé (*pin-hole*).

4.3 Volume de reconstruction

Une fois la calibration effectuée, il reste à déterminer, dans l'espace tridimensionnel calibré, un volume englobant la scène à reconstruire, que nous appelons *volume de reconstruction*. Nous choisissons typiquement une primitive simple. Par *primitive*, il est question d'une forme géométrique quelconque, tel un cube, une sphère ou un cylindre. Pour la définition du volume, n'importe quelle primitive de type polyèdre fermé ferait l'affaire.

Pour déterminer ce volume de reconstruction, nous employons une fois de plus *ImageModeler* pour calculer la position, l'orientation et la grosseur de la primitive qui sert de volume. Il procède en se servant des marqueurs (des points caractéristiques ou saillants) pour positionner la primitive. Ensuite l'usager peut la déplacer manuellement pour s'assurer que la projection de la primitive couvre la scène à reconstruire dans chaque image.

Il est possible de s'en tirer sans définir un volume de reconstruction particulier en faisant usage de l'enveloppe convexe formée par l'intersection des pyramides de vue de chaque caméra, mais seulement si les points de vue regardent la scène et sont distribués partout autour de la scène de sorte que l'intersection ne soit pas infinie. Autrement, cette intersection donnerait un volume qui s'étend à l'infini. Étant donné que ce volume sert d'espace d'échantillonnage (section 5.1), dans ce dernier cas, nous pourrions d'abord échantillonner des zones plus près des points focaux des caméras, mais il serait compliqué d'assurer un échantillonnage qui couvre la scène entière.

Pour plusieurs de nos algorithmes, nous avons également besoin de la boîte englobant le volume de reconstruction. Il s'agit d'une boîte rectangulaire alignée sur les axes X, Y et Z dont deux des sommets opposés ont respectivement comme coordonnées les valeurs minimales et maximales, selon chaque axe, des sommets du polyèdre du volume de reconstruction. Nous l'appelons la *boîte englobante*.

La figure 3.2 (a) illustre un exemple d'un volume de reconstruction avec sa boîte englobante (le tout est affiché en fil de fer). Une reconstruction partielle y est également présentée.

Chapitre 5

Génération de points

Le coeur de notre méthode pour reconstruire des scènes à partir des images calibrées et du volume de reconstruction est un processus itératif. Les étapes sont :

1. générer un certain nombre de points, où chaque génération consiste à :
   (a) générer une position tridimensionnelle aléatoirement et
   (b) déterminer si cette position peut faire partie de la scène à reconstruire ;
2. *nettoyer* l'ensemble des points générés.

Nous préconisons un échantillonnage aléatoire plutôt que structuré (par exemple une grille régulière, tel qu'employée dans d'autres algorithmes de reconstruction telles la coloration de voxels [SD99], la sculpture d'espace [KS00] ou les coupes de graphes [SVZ00]) pour les raisons suivantes :

- simplicité et généralité ;
- évitement des arrangements réguliers, auxquels notre perception visuelle est particulièrement sensible ;
- obtention d'un résultat intermédiaire plus rapidement ;
- aucun effort particulier n'est requis pour échantillonner correctement la silhouette des objets.

Il est à noter cependant que l'échantillonnage aléatoire n'est pas exclusif et que rien n'empêche d'employer des méthodes structurées. De plus, à moins d'avis contraire, par *aléatoire*, nous sous-entendons toujours une distribution uniforme, bien que rien n'empêche de faire usage d'autres types de distributions en trois dimensions (gaussienne, poisson, *etc.*). Elles sont cependant moins nécessaires dans notre contexte.

Nous emmagasinons les points dans une simple liste (un tableau à dimension dynamique). Une structure plus complexe ne s'est pas avérée nécessaire, bien que certains de nos algorithmes (dans les raffinements décrits au chapitre 6) pourraient profiter d'une

structure ayant une meilleure organisation spatiale, tel un arbre octal (*octree*) ou un arbre-kd (*kd-tree*) [Sam90].

Les détails de la génération sont décrits dans le reste du chapitre. La section 5.1 couvre les détails de l'échantillonnage aléatoire, ceux de la validation se trouvent à la section 5.2 et enfin nous parlons du *nettoyage* à la section 5.3.

5.1 Génération aléatoire

Pour générer une position pour un point, nous choisissons tout simplement une position aléatoire dans le volume de reconstruction. Cependant, étant donné que ce volume peut être un polyèdre fermé quelconque, nous employons une méthode Monte Carlo pour générer des positions uniformément distribuées dans l'espace du volume. L'uniformité est importante pour éviter une génération biaisée.

Nous générons d'abord une position aléatoire à l'intérieur de la boîte englobante du volume de reconstruction. Ensuite nous vérifions si cette position se trouve à l'intérieur du polyèdre en lançant un rayon à partir de cette position dans une direction arbitraire. Si le nombre d'intersections avec le polyèdre est impair, la position générée se trouve dans le volume et nous pouvons passer à l'étape de validation. Sinon, nous rejetons ce point et recommençons la génération.

Alternativement, au lieu d'employer le test si une position est à l'intérieur d'un polyèdre, nous pourrions utiliser la représentation implicite [Blo97] du volume de reconstruction, dans la mesure où c'est possible (pour les coniques, par exemple). À ce moment, le test d'intériorité du volume devient simplement une évaluation de la fonction implicite et de vérifier que la valeur est non positive, la génération aléatoire serait donc possiblement plus rapide.

Nous avons également étudié des méthodes plus systématiques de génération de positions. Par exemple, nous pouvons échantillonner l'espace du volume de reconstruction avec une grille régulière tridimensionnelle et perturber légèrement ces positions. Il n'y a pratiquement aucun avantage cependant à employer cette méthode comparativement à l'approche naïve décrite auparavant, car il faut malgré tout effectuer le test pour savoir si la position se trouve dans le volume de reconstruction. Si nous permettions de déborder un peu de ce volume, nous pourrions précalculer l'ensemble des cellules de la grille qui se trouvent dans le volume, mais ceci requierrait plus de mémoire. La perturbation re-

quiert aussi des évaluations d'un générateur de nombres pseudo-aléatoires. En moyenne, le gain en temps d'exécution est faible et, au niveau des résultats, étant donné que cette méthode et la précédente échantillonnent toutes les deux l'espace uniformément, il n'y a pratiquement pas de différence. Cette approche, à base de grille régulière, pourrait être tout de même plus efficace, mais elle a été jugée non nécessaire pour notre système.

Il y a d'autres méthodes de génération ayant des buts plus spécifiques. Cependant, la plupart de ces méthodes requièrent qu'un ensemble de points ait déjà été généré ou alors elles requièrent des interventions plus directes de la part de l'usager. C'est pourquoi nous les avons considérées comme étant des techniques de raffinement et elles sont décrites au chapitre 6. Voici un aperçu de ces méthodes :

- génération de points par perturbations locales autour de points déjà générés (section 6.1.1) ;
- remplissage d'un point de vue (section 6.2.2) ;
- remplissage tridimensionnel se basant sur la continuité de couleurs ou d'occupation de l'espace dans la scène (section 6.2.3) ;
- génération de points à l'aide d'information sur la géométrie de la scène (section 6.2.4).

Il serait possible aussi d'employer une méthode d'échantillonnage davantage basée sur les images d'entrée. Au lieu de définir un volume de reconstruction, nous pourrions nous servir de la silhouette de la scène dans chaque image pour générer des points dans la pyramide de chaque pixel de la silhouette, de façon similaire au remplissage de vue de la section 6.2.2. Cependant, étant donné que nous employons une représentation par points, il est important d'avoir un échantillonnage sans biais pour éviter de laisser des proportions de la scène sous-échantillonnées. Grossman et Dally [GD98] font un échantillonnage similaire mais selon des points de vue rapprochés pour s'assurer que, de tout point de vue, pour une résolution d'image donnée, il y ait au moins un point qui soit projeté dans chaque pixel. Le nombre restreint d'images à notre disposition ne permet pas ceci. De plus, leur algorithme profite du fait qu'ils échantillonnent une scène déjà construite. Ils n'ont donc pas la difficulté de devoir générer des silhouettes.

5.2 Validation

Pour chaque nouvelle position générée, peu importe la méthode employée, nous devons la valider, *i.e.* nous vérifions qu'un point à cette position puisse faire partie de la

scène à reconstruire. Pour valider un point, nous le projetons dans chacune des images calibrées (section 4.2), et nous accumulons les informations sur la couleur aux pixels correspondants. Si le point n'est pas visible pour une image donnée (section 5.2.4), alors nous ne considérons pas les informations de couleur de cette image. En d'autres mots, nous effectuons le reste de la validation avec un sous-ensemble des images excluant celle-ci. Nous obtenons ainsi une liste d'images pour lesquelles le point est considéré visible avec les couleurs des pixels associés.

Étant donné qu'un point est rarement projeté exactement au centre d'un pixel, nous considérons les couleurs des quatre pixels les plus près de la position où le point est projeté. Si la calibration de l'image n'était pas suffisamment robuste, nous pourrions considérer une région plus grande autour du point. Alternativement, nous pourrions effectuer une interpolation bi-linéaire, ce qui nous éviterait de comparer un voisinage. Cependant, nous risquons de perdre de la précision dans les zones riches en détails (textures, discontinuités). De plus, il est impossible d'employer ceci pour notre principale méthode de comparaison de couleurs (section 5.2.2), car nous devons préserver les couleurs telles qu'elles sont.

Nous nous servons de cette liste pour évaluer la "plausibilité" du point. Un point est dit plausible si une couleur est clairement dominante dans la liste. Pour trouver cette dominance, nous devons comparer les couleurs entre elles. Dans les deux prochaines sections, nous décrivons deux méthodes différentes pour comparer les couleurs (distance entre couleurs à la section 5.2.1 et quantification à la section 5.2.2). Puis nous décrivons le test de plausibilité employé dans la section suivante (5.2.3). Nous couvrons les détails de la notion de visibilité à la section 5.2.4, et enfin nous présentons une méthode pour masquer des régions d'images afin de rapidement rejeter des points non désirés à la section 5.2.5.

5.2.1 Distance entre les couleurs

Une façon de comparer les couleurs est d'évaluer la distance entre celles-ci. Nous calculons la distance Euclidienne dans un espace de couleurs, et si le résultat est inférieur à un certain seuil spécifié par l'usager, nous considérons les deux couleurs comparées comme identiques (du point de vue de la dominance).

Parmi les espaces de couleurs possibles, nous avons expérimenté avec l'espace *RGB*, CIE *xy*, CIE *Luv* et CIE *Lab* [WS82]. L'espace *RGB* est employé par les cartes vidéo,

il devrait donc être plus efficace que les autres (aucun passage d'un espace à un autre n'est requis). L'espace CIE *xy* est un espace de chromaticité où il n'y a pas de notion d'intensité. Il devrait permettre de réduire les effets d'éclairage. Les espaces CIE *Luv* et CIE *Lab* sont des espaces dont la distribution des couleurs tend à être plus uniforme au niveau de notre perception humaine des couleurs. Ainsi, si la distance entre deux paires de couleurs est la même, nous nous attendons à ce que perceptuellement les couleurs d'une paire diffèrent entre elles à peu près autant que les couleurs de l'autre paire. Étant donné que nous employons un seuil fixe pour la comparaison, ces espaces devraient nous donner des résultats perceptuellement plus cohérents.

Parmi tous ces espaces, pour les fins de notre système, aucun d'eux ne s'est clairement démarqué comme étant le meilleur pour la comparaison. L'usage de *RGB* est bel et bien plus rapide, puisque les données originales sont exprimées en *RGB*. Ainsi, plus de points peuvent être examinés, ce qui compense pour la comparaison de couleurs moins précise dans cet espace. Pour limiter les conversions requises des autres espaces, il serait possible de les précalculer en convertissant chaque image, mais étant donné que nous avons toujours besoin des images en *RGB* pour l'affichage, ceci requerrait le double des images en terme de mémoire. La conversion de la couleur dominante en *RGB* serait toujours nécessaire. En dehors de ces considérations, nous avons observé que les espaces CIE *xy*, CIE *Luv* et CIE *Lab* se comportaient très similairement.

5.2.2 Quantification

Une alternative à la méthode précédente est de quantifier les images d'entrée au niveau des couleurs, *i.e.* réduire le nombre de couleurs présentes dans les images, et alors comparer les couleurs en testant leur égalité. Ainsi, nous remplaçons le seuil de comparaison de la méthode précédente par un seuil global et fixe. Cette méthode de comparaison est beaucoup plus rapide et étant donné que, lors d'une séance de modélisation, plusieurs millions de points sont appelés à être examinés, cette rapidité est fort souhaitable. Cependant, la quantification est coûteuse, il ne faut donc pas la refaire souvent lors d'une séance de travail pour profiter du gain d'efficacité à la comparaison. Idéalement, nous l'effectuons une seule fois sur les images en précalcul. Nous pouvons ainsi nous permettre d'employer des méthodes de quantification plus sophistiquées, qui pourraient traiter les couleurs des pixels adjacents, des critères perceptuels, *etc.*

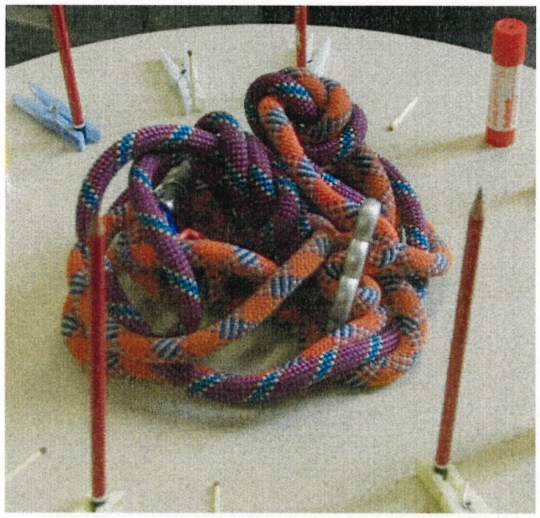
Enfin, pour la quantification, nous ne considérons que les pixels qui se trouvent à

l'intérieur de la projection dans chaque image du volume de reconstruction. Ceci permet de mettre une emphase sur les couleurs de la scène à reconstruire et non sur ses environs. Nous avons également observé que la quantification tend à réduire les effets d'éclairage dans les images.

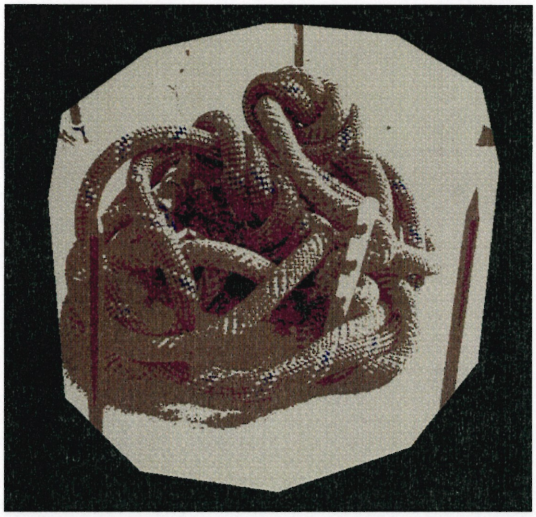
Dans notre système, l'usager spécifie le nombre maximum de couleurs désirées pour représenter toutes les images quantifiées, *i.e.* toutes les images quantifiées partagent le même ensemble de couleurs. Il le faut bien, car autrement le test d'égalité ne serait plus valide d'une image à l'autre. Cette approche requiert une bonne compréhension de la distribution des couleurs dans les images afin de choisir le bon nombre de couleurs. Comme nous l'avons mentionné, comparer les couleurs ainsi revient à peu près à fixer un seuil global pour la comparaison des couleurs. Ainsi, plus le nombre de couleurs est faible, plus ce seuil est grand, il y aura donc ainsi davantage de points qui seront acceptés. Cependant il y a un fort risque que plusieurs de ces points n'auraient pas été acceptés en employant la première méthode de comparaison de couleurs, produisant ainsi une reconstruction moins exacte. En pratique, dans l'ensemble de nos tests, nous avons spécifié un nombre de couleurs variant entre 10 et 64. En général, 10 ou 16 couleurs ont suffi. Il est important que l'usager regarde les images quantifiées produites pour vérifier si le résultat est visuellement satisfaisant. Les caractéristiques de la scène doivent demeurer nettement distinguables. Il faut aussi noter que la plupart des dégradés se transforment en bandes de couleurs uniformes. Les frontières de ces bandes peuvent poser des problèmes, mais nous n'en avons pas observé en pratique.

La figure 5.1 donne un exemple d'une des images employées pour la scène des cordes de l'exemple du chapitre 3. À seulement 5 couleurs, nous perdons beaucoup de détails de la scène et ceci résulte en beaucoup de points qui n'auraient pas dû être acceptés (figure 8.3 (a), section 8.1.2). Par contre, à 10 couleurs nous pouvons déjà bien discerner les détails et nous obtenons de bons résultats (figure 8.3 (b)). À 30 couleurs, les détails sont encore plus saillants, mais les résultats sont comparables à l'utilisation de 10 couleurs seulement (figure 8.3 (c)), quoique certaines régions de la scène deviennent plus difficiles à reconstruire.

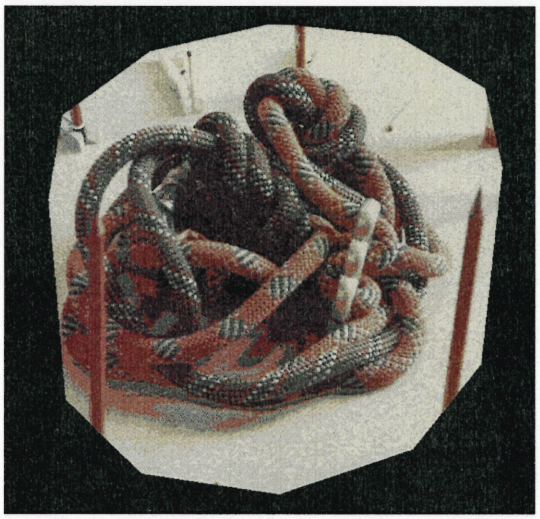
De plus, nous avons mentionné pour la méthode de comparaison précédente qu'il est possible de précalculer la conversion de couleurs pour les images mais que ceci impose un certain coût en terme de mémoire (double de l'espace requis pour les images). Le même



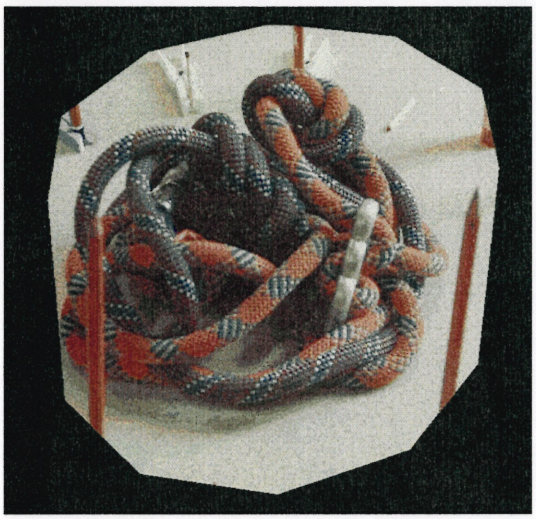
(a) Image originale



(b) Quantification à 5 couleurs



(c) Quantification à 10 couleurs



(d) Quantification à 30 couleurs

FIG. 5.1 – Effet de la quantification.

problème se pose ici, mais étant donné que le nombre de couleurs varie généralement entre 16 et 128, il est donc possible d'encoder les images quantifiées par index de couleurs avec seulement 8 bits par pixel, au lieu de 24 bits (en supposant que chaque composante de couleur prend 8 bits, comme c'est généralement le cas). Ainsi, nous prendrions seulement le tiers plus de mémoire pour les images quantifiées. Nous pourrions même employer directement cet index pour la comparaison.

5.2.3 Le test de plausibilité

Maintenant que nous pouvons comparer les couleurs, il nous reste à déterminer la dominance d'une couleur dans la liste en provenance des projections d'un point candidat. Nous comparons tout d'abord chaque échantillon de couleur en employant l'une des méthodes de comparaison décrites précédemment. Par échantillon, il est question des quatre pixels voisins du point exact où est projeté le point tridimensionnel dans une image. L'échantillon qui obtient le plus de comparaisons positives est considéré comme dominant, et sa couleur (dans l'image originale) est attribuée au point.

Lorsque nous comparons les échantillons entre eux, dans le cas d'une comparaison par distance entre couleurs, nous ne comparons que le pixel dans lequel le point est projeté. Dans le cas d'une comparaison par quantification, nous comparons le pixel dans lequel le point est projeté avec les quatre pixels de l'autre échantillon. La comparaison est positive si l'une des quatre comparaisons l'est.

La relation entre la taille de la liste et le nombre de comparaisons positives est appelée la plausibilité du point. Si cette dernière valeur est supérieure à un seuil défini par l'usager, le point est dit plausible. Une valeur typique pour ce seuil est de 80%.

La figure 5.2 (a) illustre le test de plausibilité. Nous avons ici quatre caméras autour de la scène des cordes de l'exemple du chapitre 3. Le point A, qui est sur l'objet, est visible dans chaque image et toutes ses projections ont la même couleur, lui donnant une plausibilité de 100%. Le point B, également sur l'objet, est caché dans deux caméras (3 et 4). Cependant, sans information de visibilité, il y a tout de même une couleur dominante, donnant une plausibilité de 50% au point. Enfin, le point C ne se trouve pas sur l'objet et ses quatre projections donnent quatre couleurs différentes, donnant ici une plausibilité de 25%.

L'usager peut également spécifier une taille minimale pour la liste. Imposer une taille minimale implique que dès la projection du point, si la liste obtenue est inférieure

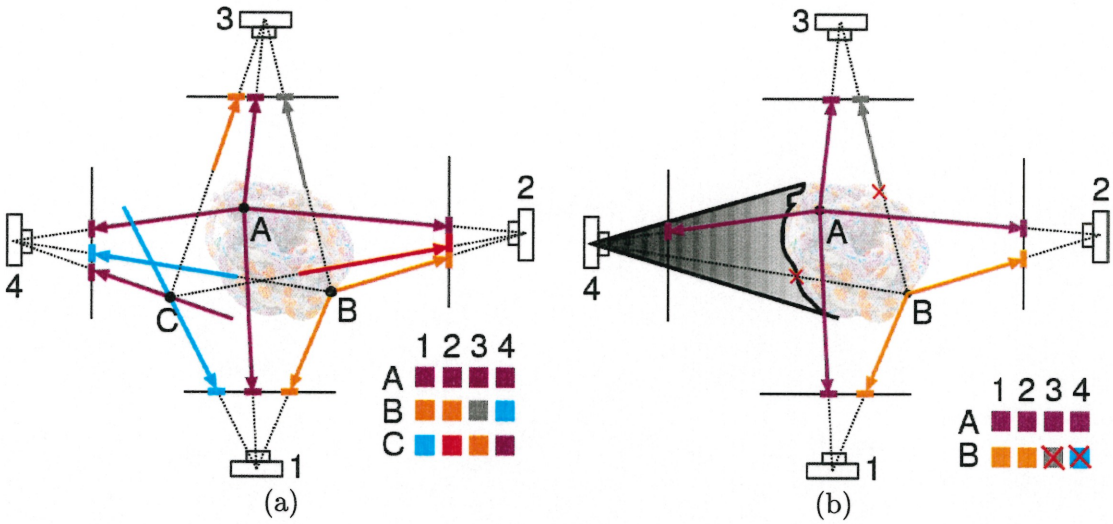


FIG. 5.2 – Test de plausibilité pour les points A, B et C.

à cette taille, le point est automatiquement rejeté. Ceci permet d'éviter des positions arbitraires d'un point lorsqu'il est visible dans une seule image (avec une taille minimale de deux ou plus).

5.2.4 Visibilité

Nous avons mentionné que la liste est construite à partir des projections obtenues d'un point pour lesquelles ce dernier est visible. Pour déterminer cette visibilité, nous procédons à l'aide de cartes de profondeurs associées à chaque image. Dans chaque carte de profondeurs est inscrite la profondeur d'un point projeté dans un pixel donné de l'image. Si un point est projeté dans un pixel pour lequel un autre point a déjà été projeté, il est considéré comme visible seulement si sa profondeur est plus petite que celle déjà inscrite. Autrement, il est considéré comme étant caché et l'image concernée est exclue de la liste. Pour éviter qu'un point se cache lui-même, ainsi que d'autres points qui sont situés très près les uns des autres, un biais, dont la valeur est spécifiée par l'usager, est ajouté à la profondeur d'un point lorsqu'il est inscrit dans la carte de profondeurs de chaque image.

Si nous revenons à l'exemple de la figure 5.2, sans information de visibilité, le point B sera rejeté, à moins de spécifier un seuil très bas (50% ou moins). Cependant, avec cette information (figure 5.2 (b), où nous illustrons en noir la carte de profondeurs pour la caméra 4), puisque le point est occulté pour les caméras 3 et 4, nous ne considérons que la liste des projections obtenues des caméras 1 et 2, sa plausibilité passe alors à 100%.

Idéalement, les cartes de profondeurs devraient être mises à jour à chaque fois qu'un point est généré. Cependant, il est possible de profiter de l'accélération matérielle des cartes vidéo pour rapidement générer ces cartes de profondeurs pour l'ensemble des points. Ainsi, au lieu de mettre à jour les cartes de profondeurs à chaque point généré, nous le faisons lorsque nous avons un certain nombre de points qui ont été générés et validés.

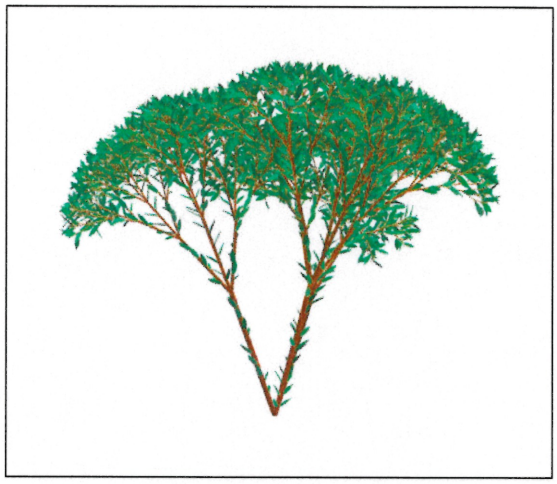
5.2.5 Masques de rejet

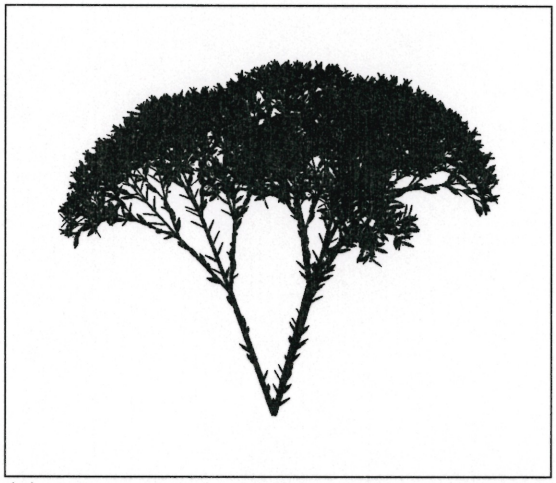
Un autre outil disponible à l'usager permet d'exclure des points qui seraient projetés dans un endroit hors de la scène véritable dans une image. Pour chaque image, l'usager peut spécifier un masque binaire (une image de même dimension que l'image d'entrée dont chaque pixel est un bit). Lorsqu'un point est projeté dans une image et qu'à la même position dans le masque la valeur n'est pas zéro, le point est automatiquement rejeté (il n'est même pas considéré comme caché, il est considéré comme s'il avait échoué son test de plausibilité). Sinon, nous procédons avec le point normalement.

L'usage de tels masques permet d'éliminer les points qui seraient produits à partir du fond de la scène et d'accélérer la génération de points en rejetant rapidement les points qui, pour la plupart, auraient de toute façon été rejetés par le test de plausibilité. Ceci est particulièrement pratique dans les cas où le fond est d'une couleur uniforme, ce qui est souvent le cas des images de synthèse. Dans un tel cas, tous les points projetés dans le fond dans chaque image seraient acceptés alors qu'ils ne sont pas désirables. Il est facile de produire un masque en effectuant quelques substitutions de couleurs à partir de l'image originale, supposant que le fond ait une couleur différente. La figure 5.3 (a) montre un exemple d'une telle situation (l'ensemble des images de la scène se trouve à la figure 8.6, section 8.2).

Il n'est pas toujours facile d'automatiquement générer des masques. Dans ce cas, l'usager peut peindre le masque en superposition avec l'image d'origine. Dans l'état actuel du système, nous n'avons pas implanté de méthode pour générer les masques à partir du système lui-même. L'usager doit les créer avec d'autres outils (tels *Gimp* [Gim] ou bien *Photoshop* de *Adobe* [Ado]) et fournir les images binaires en plus des images d'entrée. La figure 5.3 (b) montre un exemple d'un masque créé avec *Gimp* (l'ensemble des images de la scène se trouve à la figure 8.8, section 8.3).

Nous pouvons également nous servir de ces masques pour aider davantage la quan-





(a) Masque d'une image d'une scène dont la couleur de fond est uniforme. Le masque est généré automatiquement par substitution de couleurs dans l'image d'origine.

(b) Masque créé manuellement avec *Gimp* pour la scène de la cafetière (section 8.3). Le masque ici a été utile pour accélérer la génération de points car le volume de reconstruction était plus grand qu'il ne le fallait.

FIG. 5.3 – Exemples de masques de rejet.

tification (section 5.2.2). Pour chaque image, il suffit de produire un masque correspondant à l'union du masque fourni et du masque produit par la projection du volume de reconstruction, et de considérer seulement les pixels qui ne sont pas masqués.

5.3 Nettoyage

Dans les premières phases d'une séance de modélisation, des points faux peuvent être générés. Un point faux est un point qui a passé le test de plausibilité mais qui n'est pas un point faisant véritablement partie de la scène, tel un point qui ne se situe pas à la surface de l'objet auquel il est associé, dû à de multiples positions possibles lors de sa génération. Les points faux sont fréquemment le résultat d'un manque d'information sur la visibilité. La figure 5.4 illustre un exemple d'un point faux. Du point de vue de la caméra 1, il n'y a pas de différence entre les points R, P et S. Si ces trois points projettent dans des pixels de l'image de la caméra 2 ayant une couleur *comparable*, ils peuvent tous les trois être considérés plausibles. Cependant, le point P est un point faux, *i.e.* il se trouve à l'intérieur de l'objet, et il est possible qu'il soit ajouté s'il est généré en premier.

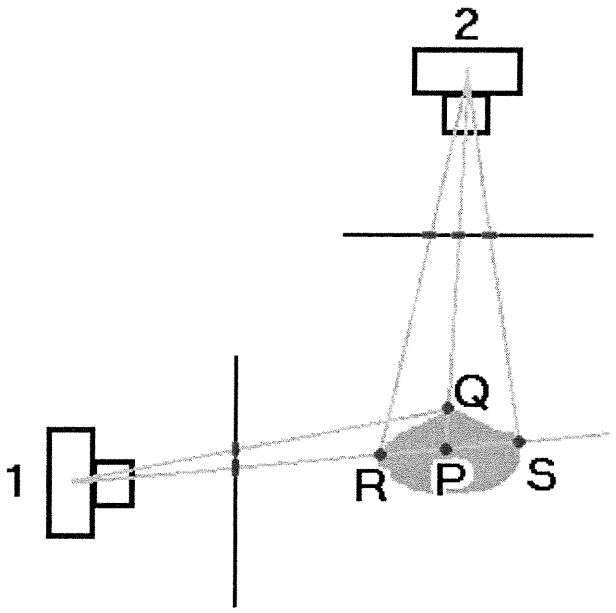


FIG. 5.4 – Point faux.

Nous présentons deux méthodes pour *nettoyer* ces points, *i.e.* les retirer de l'ensemble des points générés. Elles consistent à retester les points avec des cartes de profondeurs mises à jour (section 5.3.1) ou à reprojeter les points dans les images visibles pour en

comparer les couleurs (section 5.3.2).

5.3.1 Revalidation

Avec la génération de nouveaux points, nous avons plus d'information sur la visibilité de la scène et il serait possible d'automatiquement rejeter une bonne partie des points faux, *i.e.* ceux qui ne sont plus visibles dans aucune image. Il est également possible, en cours de route, de vouloir changer certains seuils, tel le seuil pour le test de validité ou alors le nombre minimum ou maximum d'images. Dans tous ces cas, nous pouvons lancer une revalidation des points. Il s'agit de repasser chaque point généré dans le processus de validation et nous conservons seulement ceux qui passent le test de validité.

Ainsi, pour revenir à l'exemple de la figure 5.4, après que les points R et Q aient été générés, le point P n'est plus visible dans les deux caméras. En revalidant les points, il serait donc exclu.

5.3.2 Par rendu

Alternativement, il est possible de nettoyer les points en comparant directement leur projection dans chaque image. Pour le point de vue de chaque image, nous effectuons un rendu (chapitre 7) des points et nous comparons l'image obtenue avec l'image originale. Tout point qui a une différence significative avec la couleur du pixel où il projette dans l'image d'origine est exclus. Cette méthode de nettoyage assure une cohésion plus locale à chaque image avec les points générés.

Afin de rapidement retrouver quel point a été projeté dans quel pixel, nous faisons usage d'une carte d'identités, *i.e.* dans une image annexe, nous projetons le point avec une couleur unique, son identificateur. Cette dernière est déterminée en découpant en trois morceaux de 8 bits les 32 bits servant à représenter l'index du point dans son ensemble. Nous avons une perte de 8 bits en effectuant ceci, mais il est rare que nous ayons besoin de plus de 24 bits pour adresser les points (avec 24 bits, nous pouvons avoir un ensemble d'au plus 16 777 215 points, ce qui est amplement suffisant, considérant qu'en général nous avons un modèle de bonne qualité avec 500 000 points). Chaque morceau est attribué à une des composantes rouge, vert ou bleu de la couleur. Pour retrouver le point, il suffit de recoller les morceaux.

Nous pourrions tout simplement prendre chaque point individuellement et comparer sa projection dans chaque image, en n'oubliant pas de tenir compte de sa visibi-

lité dans la carte de profondeurs, quoique s'il n'est pas visible, il ne faut pas faire la comparaison pour cette image. Cependant, avec la première approche, nous pouvons exploiter l'accélération matérielle des cartes vidéo et avoir un gain substentiel en vitesse d'exécution.

Chapitre 6

Techniques de raffinement

Le processus de génération décrit au chapitre 5 suffit à lui-même pour produire des résultats satisfaisants pour plusieurs applications en infographie (les figures 8.3 (b), (c) et (d), 8.7 (f) et 8.9 (a) et (b) en montrent des exemples). Cependant, il est possible de fournir des outils à l'usager pour soit accélérer la génération de nouveaux points dans des régions d'intérêt, soit améliorer la qualité visuelle de l'ensemble actuel de points.

Nous décrivons dans ce chapitre deux classes de ces outils disponibles à l'usager : automatique (section 6.1) et interactif (section 6.2). Les outils de raffinement automatique agissent sur l'ensemble des points et ne requièrent aucune intervention de l'usager, si ce n'est de lancer l'exécution de l'outil ou de spécifier quelques paramètres, tel le seuil de comparaison de couleurs pour le test de validité. De leur côté, les outils interactifs exigent plus d'intervention de l'usager au niveau de la localité des opérations ou de ses connaissances de la scène à reconstruire. Dans tous les cas, sauf exception, ces outils requièrent qu'un ensemble de points ait déjà été généré.

6.1 Raffinement automatique

Les techniques de raffinement automatique améliorent l'ensemble de points de façon globale. Ainsi, l'outil agit sur tous les points de l'ensemble sur lequel il est appliqué. Cet ensemble correspond typiquement à la totalité des points générés, mais rien n'empêche de restreindre une opération dite globale sur un sous-ensemble de ces points. Cependant, comme nous n'avons pas de moyen automatique pour déterminer de tels sous-ensembles, nous avons laissé ce soin à l'usager par des techniques interactives (section 6.2.1).

Nous présentons deux outils qui se sont avérés efficaces : la perturbation de points (section 6.1.1) et la fusion de points (section 6.1.2). Avec la perturbation de points (*jitter*), nous tentons de déplacer légèrement chaque point en espérant que sa nouvelle

position soit meilleure, dans le sens de la valeur du test de plausibilité. Nous pouvons également nous servir de cet outil comme méthode de génération de points, en conservant tous les points ainsi perturbés. De son côté, la fusion cherche à réduire les redondances dans une région trop dense en points en remplaçant plusieurs points par un seul.

6.1.1 Perturbation

Lorsqu'un point est généré, sa position actuelle n'est pas nécessairement la meilleure en terme de plausibilité ou elle peut être fausse, résultant de multiples positions possibles. Cependant, en général, si un point est généré à une position donnée, nous pouvons nous attendre à ce que son voisinage immédiat soit propice à générer des points ayant une plus grande plausibilité.

Ceci amène l'idée de perturber la position d'un point, *i.e.* le déplacer à l'intérieur d'un petit voisinage, dont la taille est spécifiée par l'usager. Ensuite, nous pouvons vérifier si sa plausibilité est plus grande à la nouvelle position. Si tel est le cas, nous déplaçons le point à cette position et lui attribuons la nouvelle couleur, espérant ainsi converger vers une meilleure solution pour la reconstruction de notre scène. Si la plausibilité n'est pas meilleure, nous ignorons la perturbation et gardons le point à sa position initiale. Nous pouvons ainsi appliquer ce procédé sur tous les points de l'ensemble considéré et le répéter jusqu'à ce que la majorité des perturbations soient ignorées ou jusqu'à ce que l'usager décide de l'arrêter.

Lors d'une séance de modélisation, appliquer la perturbation sur tous les points générés peut nuire à l'interactivité du système si le nombre de points est très grand. Nous pourrions tout simplement permettre d'interrompre le processus, mais ceci ferait en sorte que seulement les premiers points générés auraient une chance d'être perturbés, ou alors ils seraient perturbés plus souvent que les autres pour plusieurs itérations du processus. Pour éviter cette situtation, nous permettons également de perturber un sous-ensemble de points choisis aléatoirement parmi tous les points générés. En moyenne, après un certain nombre d'itérations, tous les points devraient avoir été perturbés au moins une fois, et ce, sans biais par rapport à l'ordre de génération.

Nous avons cependant observé en pratique que cette perturbation n'a pas d'impact majeur sur la qualité de la reconstruction lorsque de nombreux points ont déjà été générés (section 8.1.3).

Génération de points par perturbation

De cette idée de perturbation dans le voisinage d'un point découle une méthode de génération de points. Étant donné que les points générés dans le voisinage d'un autre point ont une plausibilité au moins aussi bonne que ce point, il est donc intéressant de se servir de ce point comme germe de génération par perturbation. Au lieu de déplacer le point suite à une perturbation comme décrite ci-avant, nous gardons simplement l'ancien point et ajoutons le point perturbé à l'ensemble des points générés.

Cependant, pour que cette méthode de génération donne de bons résultats, il faut que l'ensemble de points initiaux à partir desquels nous lançons ce processus soit une bonne approximation de la scène à reconstruire, car procéder ainsi engendre nécessairement un biais en faveur du voisinage des points actuels. Autrement dit, le reste de l'espace n'est pas considéré comme dans le cas de la génération de base. Il est possible de réduire l'impact de ce biais en agrandissant le voisinage autour de chaque point pendant un certain nombre d'itérations, jusqu'à ce que l'ensemble de points soit plus représentatif de la scène. Ainsi, dans l'exemple du chapitre 3, pour obtenir le résultat de la figure 3.2 (b), nous avons attendu d'avoir obtenu un ensemble de points donnant suffisamment d'information sur la visibilité et suffisamment représentatif de la scène pour faire usage de la génération de points par perturbation. Cependant, pour éviter de pénaliser les régions non encore couvertes de la scène et de sur-échantillonner les régions déjà bien couvertes, nous avons continué d'employer la génération de base et nous avons employé la génération par perturbation seulement à partir des points générés à l'étape précédente.

Nous pouvons fractionner ce processus sur des sous-ensembles de points exactement comme pour la perturbation simple des points.

Un inconvénient de cette méthode de génération est que les nouveaux points ont une bonne probabilité de se trouver près des autres engendrant ainsi de la redondance. Il serait possible d'employer des techniques de distributions de Poisson pour éviter cela. Cependant, nous n'avons pas rencontré de problème particulier relié à la distribution uniforme aléatoire. Il y a toujours l'outil de fusion (section 6.1.2) pour aider à réduire les redondances.

6.1.2 Fusion

Étant donné que nous générons des points en échantillonnant l'espace de façon aléatoire, soit par la génération de base, soit par l'intermédiaire d'une des méthodes

de raffinement, il est possible d'avoir comme résultat des petites régions de l'espace très denses en points, et ainsi il peut y avoir beaucoup de redondance. Pour pallier ce phénomène, l'usager peut réduire l'ensemble de points par l'intermédiaire de regroupements de points. Les points qui sont situés à au moins une distance donnée (spécifiée par l'usager) les uns des autres forment un groupe. Chaque groupe est ensuite remplacé par son ou ses meilleurs représentants selon la distance et la distribution des couleurs dans ce groupe.

Une implantation simple place les points dans une grille de voxels de résolution spécifiée par l'usager et délimitée par la boîte englobante du volume de reconstruction. Pour chaque voxel, nous conservons le point ayant la plus grande plausibilité. Alternativement, nous pourrions fusionner les points du voxel selon divers critères (distribution spatiale, couleurs, *etc.*). Cette méthode a l'inconvénient de consommer beaucoup de mémoire si la résolution de la grille est très fine. Cependant, si l'ensemble de l'espace n'est pas densément échantillonné, nous pouvons nous en tirer en effectuant un adressage dispersé (*hashing*) de voxels. Il faut aussi noter qu'avec cette implantation simple, il peut encore rester des redondances après cette fusion si les points conservés de voxels voisins se situent près les uns des autres à proximité des frontières communes des voxels. En effet, la grille de voxels ne tient pas compte de la distribution spatiale exacte des points. La figure 3.3 (e) de l'exemple du chapitre 3 montre qu'il est possible d'effectuer une fusion avec cette méthode sans affecter grandement l'aspect visuel de l'ensemble des points, quoique nous ayons employé une résolution de grille assez fine (293 par 224 par 256 voxels).

Pour effectuer une réduction de l'ensemble de points plus optimale, nous devrions adapter des méthodes de simplication pour l'affichage à de multiples résolutions de maille telle celle de Rossignac et Borrel [RB93]. D'autres méthodes se rattachant plus à la représentation par points pourraient également être employées [RL00, Lin01].

6.2 Raffinement interactif

L'ensemble des méthodes automatiques décrites dans la section précédente ne requière de la part de l'usager que la décision de les employer. Il serait même possible d'établir des heuristiques pour les enclencher automatiquement. Ces méthodes ont aussi un effet global sur l'ensemble de points sur lequel elles travaillent.

Il peut être souhaitable d'avoir plus de contrôle sur la production ou la manipulation des points générés. Par exemple, l'usager pourrait souhaiter générer plus de détails dans des régions de la scène où il y a beaucoup d'occultation, par exemple le feuillage d'une plante ou encore près de fines structures (fils, tiges, *etc.*). Dans ces situations, le processus de génération de base prendrait beaucoup de temps pour en arriver au niveau de détail désiré. Les sections 6.2.1 et 6.2.2 montrent deux exemples d'outils fournisssant ce type de contrôle. L'un est un outil pour effectuer la plupart des opérations plus automatiques déjà décrites, mais dans une région délimitée par l'usager. L'autre permet de générer des points pour remplir autant que possible le point de vue actuel.

Dans la section 6.2.3, nous présentons un autre type d'outil donnant un contrôle sur la génération plus localisée de points. L'usager peut lancer un remplissage tridimensionnel (*flood fill*) se basant sur un ensemble cible de couleurs, également spécifiées par l'usager.

Finalement, étant donné que notre méthode de génération de points converge généralement vers l'enveloppe photographique (chapitre 9), nous pouvons tirer profit des connaissances géométriques de la scène par l'usager pour mieux en cerner la forme et réduire les ambiguïtés. La section 6.2.4 décrit une méthode de manipulation et de génération des points à partir de primitives.

6.2.1 Opérations locales

L'ensemble des opérations décrites jusqu'à présent s'applique soit sur l'ensemble des points ou alors sur l'espace entier du volume de reconstruction. Cependant, la scène à reconstruire n'occupe pas nécessairement l'entièreté de cet espace, et un échantillonnage uniformément spatialement distribué de la scène peut engendrer une convergence lente pour les régions avec beaucoup de discontinuités, de variations de couleurs ou de structures fines. Un outil pour effectuer des opérations dans une plus petite région du volume de reconstruction ou sur un sous-ensemble des points est alors un atout important.

Pour parvenir à cette fin, nous avons défini un outil pour positionner une primitive dans l'espace et effectuer des opérations à l'intérieur de celle-ci. Nous appelons *région d'intérêt* cette primitive et l'espace qu'elle occupe. Nous donnons ici les détails de la manipulation de cette région et des opérations effectuées avec la région dans les sections qui suivent. La figure 3.2 (c) et (d) et la figure 3.3 (h) illustrent l'usage de la région d'intérêt.

Positionnement

Typiquement, pour placer une primitive dans l'espace tridimensionnel, nous lui donnons sa position à l'aide de vues multiples ou en bougeant le point de vue (caméra), mais ceci est parfois difficile. Dans notre cas, nous avons souvent plusieurs points déjà dans la région où l'usager veut placer la primitive. Nous pouvons donc les utiliser pour un positionnement plus efficace, similairement à un pinceau en trois dimensions [HH90, ZPKG02].

L'usager n'a qu'à placer le curseur de la souris vis-à-vis d'un des points affichés et la région sera automatiquement centrée dans l'espace à la position du point tridimensionnel correspondant. Pour vérifier qu'il y a bel et bien un point sous le curseur, nous consultons le contenu de la carte de profondeurs à sa position. Si la profondeur lue est plus petite que la profondeur du fond, c'est donc qu'un point a été dessiné à cet endroit. Pour retrouver la position tridimensionnelle de ce point, il suffit de prendre la coordonnée en espace image du point (*i.e.* les coordonnées x et y du pixel avec la profondeur lue) et de la multiplier par l'inverse de la matrice de projection du point de vue courant. Alternativement, nous pourrions faire usage de la carte d'identités employée pour le nettoyage par rendu (section 5.3.2), mais ceci requiert une passe de rendu supplémentaire pour générer la carte.

S'il n'y a aucun point dans la région désirée, nous pouvons générer plus de points et attendre qu'il y en ait au moins un qui apparaisse au bon endroit. Ceci n'est pas garanti, mais dû à l'échantillonnage uniforme du volume de reconstruction, en pratique, un tel point apparaît dans un délai de quelques secondes. Alternativement, nous pourrions générer des points le long du rayon partant du point de vue et passant par le pixel sous le curseur de la souris, similairement au remplissage de vue (section 6.2.2). Enfin, nous pourrions nous contenter de la méthode typique de positionnement manuel avec vues multiples.

Forme

La forme exacte de la région, *i.e.* le type de primitive employée, peut être complètement arbitraire, de même que sa taille. Il peut s'agir d'une boîte rectangulaire, d'un ellipsoïde, d'un polyèdre, d'un cône, *etc.* Nous pourrions fournir à l'usager un choix parmi une banque de primitives. En pratique, nous nous sommes contentés d'une sphère de rayon variable, contrôlée par l'usager. Étant donné qu'il est souvent question de voisi-

nage lors d'opérations locales, cette primitive correspond à la forme le plus intuitivement liée à cette notion.

Cependant, des formes plus arbitraires seraient certainement un atout, en particulier pour cerner des régions qui ne sont en général pas sphériques, mais qu'il serait pratique d'avoir pour accélérer la génération de points dans des régions difficiles (par exemple, avec beaucoup de discontinuités tel un feuillage d'une plante) qui sont défavorisées par l'échantillonnage uniforme de la méthode de base.

Il serait également possible de se servir des points voisins du centre de la région d'intérêt pour en donner la forme ou en ajuster la position. Par exemple, nous pourrions nous servir de la distribution des points pour orienter un ellipsoïde. Ceci est utile pour aider à remplir des surfaces avec des ellipsoïdes plats. Nous pourrions aussi nous servir de ces points pour placer un cylindre ou un cône dans les régions où il y a beaucoup d'occultations ou de structures fines.

Opérations

Voici une description d'opérations possibles avec la région d'intérêt.

Sous-volume de reconstruction La région d'intérêt permet de restreindre les opérations ayant une portée globale sur l'espace ou les points. Ainsi, au lieu de générer des points en échantillonnant aléatoirement l'espace entier du volume de reconstruction, nous pouvons échantillonner seulement la région d'intérêt. Étant donné que la forme de la région peut être tout aussi arbitraire que celle du volume de reconstruction, nous procédons aussi avec une méthode Monte Carlo pour échantillonner aléatoirement et uniformément la région d'intérêt. Il est à noter qu'en plus de vérifier si le point se trouve dans la région d'intérêt, il faut aussi vérifier qu'il se trouve dans le volume de reconstruction, car rien ne garantit que la région soit entièrement incluse dans le volume. Nous pourrions effectuer un test pour savoir si la région d'intérêt est entièrement incluse dans le volume de reconstruction, et si tel est le cas, nous n'aurions pas à faire deux vérifications. Il est à noter aussi qu'au lieu de nous servir du volume de reconstruction défini par la région d'intérêt, nous pourrions nous restreindre à sa surface.

Sélection de points Pour les opérations qui s'appliquent sur les points générés, nous nous servons de la région d'intérêt pour sélectionner le sous-ensemble de points sur lequel travailler. Ce sous-ensemble correspond à tous les points qui se trouvent dans

la région d'intérêt. Les opérations principalement concernées sont le nettoyage (section 5.3) et les raffinements automatiques (section 6.1). Rien n'empêche d'aller au-delà de la sélection des points qui se trouvent dans la région d'intérêt courante. Nous pouvons nous servir de l'outil pour faire une sélection incrémentielle en accumulant les points d'une région à l'autre. Il est ainsi possible d'avoir une sélection de forme arbitraire. Ce mode de sélection est nécessaire pour les manipulations géométriques de la section 6.2.4. Il est possible ici aussi de se servir seulement de la surface de la région d'intérêt pour sélectionner des points.

Extraction d'informations Il est possible de se servir de la région d'intérêt pour extraire des informations sur les points qu'elle renferme (ou alors sur les points de la sélection). Nous nous servons entre autres des couleurs des points pour le remplissage tridimensionnel (section 6.2.3).

Suppression de points Finalement, nous pouvons nous servir de la région d'intérêt pour effectuer un nettoyage manuel, *i.e.* enlever les points qui se trouvent dans la région. Il arrive souvent que des points à des positions aberrantes soient générés. Ceci est souvent dû à des imprécisions dans les entrées, à un nombre insuffisant d'images pour bien capter la forme d'une région, à une quantification trop forte (peu de couleurs), à des zones de couleurs uniformes ou alors à des seuils pas assez stricts. Nous pouvons nous servir de cet outil pour enlever manuellement ces points.

6.2.2 Remplissage de vue

Un autre type de contrôle sur la localité de génération de points, qui est de nature très différente de la localité de l'outil de la section 6.2.1, est de générer des points de telle sorte que le rendu (chapitre 7) couvre autant que possible le point de vue actuel. Cette méthode génère rapidement une vue plus riche en détail, et si elle est employée sur plusieurs points de vue distribués uniformément autour de la scène, elle peut produire un ensemble de points adéquats pour permettre un usage efficace des autres méthodes de raffinement, en particulier la perturbation (section 6.1.1) et l'usage de notions géométriques (section 6.2.4). Il est à noter qu'il s'agit d'une des quelques méthodes qui ne requièrent pas qu'un ensemble de points ait déjà été généré.

Bien entendu, plus le point de vue se rapproche de la scène, plus nous serons en

mesure de générer plus de détails. Les images ont une résolution fixe et il y a donc une limite au niveau de détail atteignable. Passé un certain seuil, nous n'obtiendrons que des zones uniformes. Inversement, plus le point de vue est loin, moins de détails seront générés. Ceci donne un contrôle à l'usager sur le niveau de détail à atteindre.

Pour effectuer ce remplissage, il faut trouver tous les pixels dans lesquels aucun point n'est projeté (les trous). Tel qu'illustré à la figure 6.1, pour chacun de ces pixels, nous générons une position aléatoire P le long du rayon R qui part du point de vue et passant par le pixel (similairement au lancer de rayons), en se contraignant de rester dans le volume de reconstruction. Si le point à la position générée est valide, nous le conservons. Nous pouvons répéter la génération plus d'une fois si elle échoue. Le nombre d'essais maximal est donné par l'usager. Il serait possible d'ajouter une perturbation sur la direction afin de permettre de mieux échantillonner l'espace occupé par la pyramide du pixel depuis le point de vue (figure 6.2).

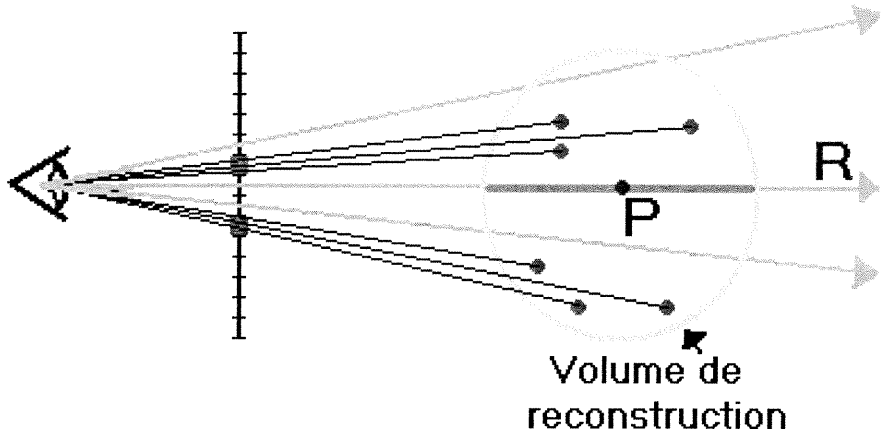


FIG. 6.1 – Génération de points par remplissage de vue.

Les figures 3.3 (f) et 3.3 (g) du chapitre 3 illustrent un exemple d'usage de cette technique pour enrichir une vue à un niveau de détail plus fin.

Nous profitons de l'accélération matérielle des cartes vidéo pour rapidement calculer les rayons de génération. Les détails des étapes du remplissage de vue sont décrits ci-après.

Espace de recherche

Étant donné qu'aucun point ne peut être généré à l'extérieur du volume de reconstruction, nous pouvons limiter notre recherche de trous dans l'ensemble des pixels

couverts par la projection du volume de reconstruction. En pratique, nous considérons un rectangle englobant cette projection pour simplifier la recherche. Ce rectangle est calculé de la même façon que la boîte englobante du volume de reconstruction (section 4.3), mais en deux dimensions et à partir de la projection des huits sommets de la boîte englobante. Bien entendu, ce rectangle ne peut pas dépasser les bornes de la clôture (*viewport*) de la fenêtre du point de vue courant, et nous effectuons un découpage (*clipping*) si cela se produit. Enfin, il est également possible pour l'usager de manuellement spécifier l'espace de recherche en choisissant interactivement avec la souris la région à remplir.

Recherche des trous

Pour trouver les trous, il nous faut un moyen pour indiquer si un point a été projeté ou non dans un pixel. Nous faisons usage du tampon stencil (*stencil buffer*). Il est possible avec ce tampon d'attribuer une valeur entière quelconque à un pixel quand un élément y est affiché (chapitre 10 de [WNDS97]). Dans notre cas, la valeur est arbitrairement choisie comme étant 1, et le tampon aura été au préalable initialisé à 0 partout. Il est alors trivial de savoir si un point a été projeté dans un pixel : il suffit de vérifier si la valeur dans le tampon est 1 ou 0. Nos pixels canditats sont tous ceux pour lesquels la valeur lue est 0.

Une autre façon serait de faire usage d'une carte d'identités telle qu'employée à la section 5.3.2. Cependant, ceci requerrait une passe de rendu supplémentaire pour les points.

Enfin, nous pourrions employer la même méthode que pour le positionnement de la région d'intérêt (section 6.2.1) avec l'aide de la carte de profondeurs. Il faudrait d'abord rendre les points et effectuer une lecture de la carte de profondeurs. À l'étape suivante, pour le rendu du volume de reconstruction tel que décrit ci-après, il faudrait désactiver le test de profondeur afin d'éviter de devoir réinitialiser la carte de profondeurs. Cette méthode est un peu plus efficace au niveau des ressources de la carte vidéo.

Calcul des rayons

Pour chaque pixel correspondant à un trou, nous devons maintenant calculer un rayon partant du point de vue et passant par ce pixel. Étant donné que nous voulons générer des points seulement à l'intérieur du volume de reconstruction, nous sommes

intéressés seulement par le segment le long du rayon qui se trouve à l'intérieur du volume. Une méthode de lancer de rayons traditionnelle ferait l'affaire, cependant nous pouvons profiter une fois de plus de l'accélération matérielle et calculer directement les segments recherchés. Il suffit d'effectuer deux passes de rendu du volume de reconstruction. À la première passe, nous n'affichons que les faces qui font dos (*backfacing*) au point de vue puis nous sauvegardons le contenu de la carte de profondeurs. À la seconde passe, nous faisons la même chose mais seulement avec les faces qui font face (*frontfacing*) au point de vue. La sélection des faces est également effectuée par le matériel de la carte vidéo (voir la section "Displaying Points, Lines and Polygons" du chapitre 2 de [WNDS97]).

Nous obtenons deux cartes de profondeurs nous donnant, pour chaque pixel, la profondeur du point le plus près et le plus loin du volume de reconstruction. Il suffit de retrouver la position tridimensionnelle du pixel à chaque profondeur (nous procédons comme pour la région d'intérêt, à la section 6.2.1) pour avoir le segment de rayon recherché. Ceci suppose que le volume de reconstruction soit convexe, ce qui est pratiquement toujours le cas (nous n'avons jamais employé des volumes concaves dans nos expérimentations). Si un volume concave est absolument requis, alors nous devrions recourir à la méthode traditionnelle de lancer de rayons.

Génération de points

Pour chaque trou, nous générons un point à une position aléatoire sur le segment puis nous le validons. Nous répétons ce processus jusqu'à ce que le test de plausibilité réussisse ou jusqu'à un nombre maximal d'essais (spécifié par l'usager).

Cette façon de procéder ne tient pas compte de la cohérence qu'il pourrait y avoir avec les points voisins. Si un bon ensemble de points a déjà été généré, nous pouvons souvent nous attendre à ce que le point le long du segment soit généré près des autres points qui ont été projetés dans des pixels voisins. Les probabilités sont fortes que ces points fassent partie de la même surface dans la scène. Avec la méthode actuelle, le point pourrait être généré beaucoup plus en avant ou plus en arrière. Dans un tel cas, au lieu de générer un point de façon uniforme sur le segment, nous pouvons le générer en fonction de la distribution spatiale des points voisins. Cette cohérence n'est pas présente pour les régions de la scène où il y a beaucoup de discontinuités, et dans cas-là, pour éviter de biaiser la génération vers de faux voisins, il est préférable d'utiliser la génération uniforme.

Cependant, sans tenir compte de la distribution des points voisins, il est possible d'aider la génération de points près de la surface la plus près du point de vue. Pour se faire, au lieu de s'arrêter au premier point plausible trouvé lors de la génération uniforme, nous poursuivons la génération, mais en ne considérant que le segment formé par ce point et l'extrémité la plus proche du point de vue. Lorsque le nombre maximal d'essais est atteint, nous conservons le dernier point généré. Nous appelons cette méthode de génération pour le remplissage de vue la *génération à proximité du point de vue.*

Avec cette méthode de génération de points le long des segments pour remplir le point de vue, nous introduisons un biais en faveur de l'espace plus près du point de vue. C'est-à-dire qu'à la base, pour chaque segment, la probabilité de générer un point près du point de vue est la même que celle pour en générer un loin. Cependant, dû à la projection en perspective, l'espace autour de la portion du segment plus proche est plus petit que l'espace autour de la portion plus éloignée, tel qu'illustré à la figure 6.2. Cependant, pour l'usage que nous faisons de cet outil, c'est même souhaitable, en particulier dans le cas où nous cherchons à générer des points près du point de vue ou lorsque nous effectuons plusieurs remplissages à divers points de vue.

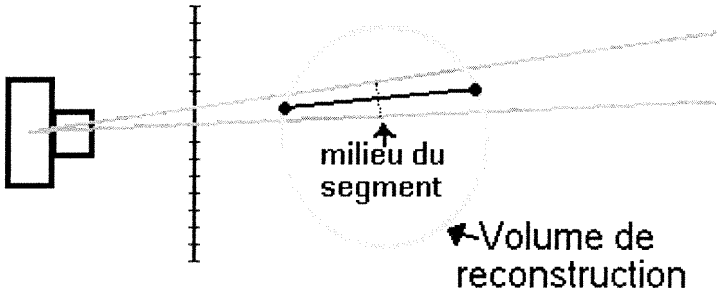


FIG. 6.2 – Distribution spatiale autour des segments de génération de points.

6.2.3 Remplissage tridimensionnel

Il arrive souvent qu'une ou plusieurs portions continues de la scène partagent des couleurs similaires, telles des régions peu texturées, ou de même illumination. Par régions continues, nous parlons de la continuité C^0 de la surface des objets de la scène. Nous pouvons exploiter ce type de cohérence pour générer plus rapidement des points qui couvrent une telle portion de scène.

Pour ce faire, nous pouvons nous servir d'un point comme germe d'un remplissage systématique de l'espace en générant des points d'abord dans un petit voisinage, puis

de plus en plus loin tant que les nouveaux points générés ont une couleur similaire à celles d'un ensemble prédéfini par l'usager.

Le choix du point germe est laissé à l'usager. Sa sélection s'effectue par l'intermédiaire de la région d'intérêt (son centre, voir la section 6.2.1). À partir de ce point, nous lançons un algorithme de remplissage tridimensionnel (algorithme 1) similaire au remplissage à germe (*flood fill*) [FvDFH96] en deux dimensions.

```
Algorithme : remplissage3D( x, y, z )
si !déjà_visité( x, y, z ) alors
    point ← générer_point( x, y, z ) ;
    si valide( point ) et couleur_similaire( point ) alors
        ajouter( point ) ;
        remplissage3D( x + 1, y, z ) ;
        remplissage3D( x - 1, y, z ) ;
        remplissage3D( x, y + 1, z ) ;
        remplissage3D( x, y - 1, z ) ;
        remplissage3D( x, y, z + 1 ) ;
        remplissage3D( x, y, z - 1 ) ;
    fin
fin
```

Algorithme 1: Remplissage tridimensionnel.

La position (x, y, z) indique un voxel dans une grille tridimensionnelle alignée et placée dans la boîte englobante du volume de reconstruction. La résolution de cette grille est spécifiée par l'usager. Il est inutile dans ce cas-ci de créer la grille tout entière étant donné que typiquement nous n'en visiterons qu'une petite partie. Ceci consommerait une très grand quantité de mémoire pour rien. Nous emmagasinons plutôt les voxels visités dans une table d'adressage dispersé (*hash table*). Ainsi, la fonction `déjà_visité` ne fait que vérifier si le voxel aux coordonnées (x, y, z) a déjà été ajouté dans la table. Les coordonnées du voxel initial sont obtenues à partir du point germe sélectionné en arrondissant au plus près.

Lorsque l'algorithme traite un nouveau voxel, il génère un point aléatoirement dans l'espace du voxel. La validation du point (la fonction `valide`) correspond à s'assurer que le point se trouve dans le volume de reconstruction et à passer le test de plausiblité. Pour ce qui est du test de similarité de couleur (la fonction `couleur_similaire`), qui est différente du test de plausibilité, nous comparons la couleur du point (selon l'une des méthodes de comparaison des sections 5.2.1 ou 5.2.2) avec les éléments d'un ensemble de couleurs préalablement défini. Il y aura similarité s'il y a une comparaison positive. Nous expliquerons plus loin comment nous définissons cet ensemble de couleurs.

Étant donné que le point est généré de façon aléatoire dans le voxel, il est possible que les deux tests échouent alors qu'ils réussiraient pour une autre position dans le même voxel (ceci est particulièrement vrai si le voxel projette dans plusieurs pixels). Pour pallier ceci, nous répétons la génération dans le voxel tant et aussi longtemps que les tests n'auront pas réussi ou qu'un nombre d'essais maximal (spécifié par l'usager) soit atteint.

En pratique, nous avons plutôt implanté l'algorithme de manière itérative à l'aide d'une pile afin d'éviter des problèmes de récursion trop profonde et de débordement de pile d'exécution (*stack overflow*). Ceci donne aussi la possibilité de momentanément suspendre l'exécution de l'algorithme sans devoir sortir de la récursion pour, par exemple, mettre à jour la carte de profondeurs.

La figure 6.3 illustre un exemple d'usage du remplissage tridimensionnel pour la scène du buisson synthétique (section 8.2). En (a), l'usager fait usage de la région d'intérêt pour donner la position du germe. Le résultat de l'exécution de l'algorithme se trouve en (b). L'ensemble des branches n'a pas été reconstruit dû aux multiples occultations causées par les feuilles.

L'ensemble de couleurs

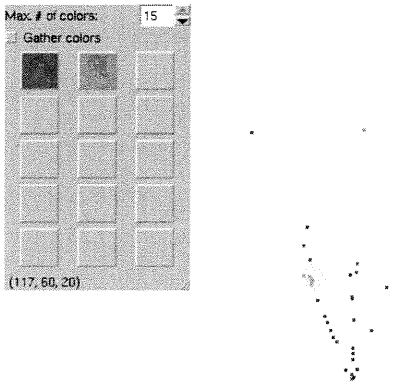
L'ensemble de couleurs est défini à l'aide de la région d'intérêt (section 6.2.1) en accumulant les couleurs des points qui se trouvent dans la région. L'usager a ensuite la possibilité de retirer les couleurs non désirées à l'aide d'une boîte de dialogue affichant les couleurs sélectionnées (figure 6.3 (a)).

Rien n'empêche d'ignorer cet ensemble. Dans ce cas-là, la seule condition d'arrêt de la propagation de la génération est le test de validité. Au lieu d'exploiter la cohérence au niveau des couleurs, nous exploitons alors la continuité d'une portion de la scène. La section 8.5 présente un exemple d'usage de ce type de remplissage.

Discontinuités

Il peut arriver qu'une région de couleur similaire de la scène soit occultée par d'autres éléments de la scène. Par exemple, les feuilles d'une plante occultent les branches. Ces occultations nuisent à la propagation de la génération. Dans de tels cas, l'usager doit lancer l'algorithme de remplissage pour chaque région séparée par les occultations.

Pour les cas d'occultation mineure, *i.e.* quand la séparation entre les régions est



(a) Ensemble de points initiaux et choix de la position du germe et des couleurs.

(b) Exécution du remplissage.

(c) Exécution du remplissage sans le test de couleur et avec un seuil réduit pour le test de plausibilité.

(d) Nettoyage des points après réduction du seuil.

FIG. 6.3 – Exemple de remplissage tridimensionnel.

petite, il y a moyen d'éviter cette approche manuelle. Une façon de faire est de permettre à l'algorithme de faire des sauts dans sa propagation. Au lieu de s'arrêter lorsque le voxel courant échoue le test de propagation (validité et couleur), l'algorithme de remplissage pourrait poursuivre sa propagation jusqu'à ce qu'il frappe à nouveau une bonne région (bonne dans le sens du test de propagation), ou jusqu'à une limite de propagation dans les zones occultées (spécifiée par l'usager en terme de niveau de récursion).

Il est toutefois possible d'aider la propagation au-delà des discontinuités sans devoir modifier l'algorithme. Il suffit de réduire le seuil de plausibilité (section 5.2.3) lors du remplissage et, une fois terminé, d'effectuer une passe de nettoyage (section 5.3) en augmentant le seuil. Ainsi, à la figure 6.3 (c), nous avons pu remplir une portion beaucoup plus grande des branches en réduisant le seuil. Nous avons aussi retiré le test des couleurs pour reconstruire les feuilles du même coup. Le haut n'est toujours pas reconstruit par manque d'information sur la visibilité. Ensuite, en (d), nous avons augmenté le seuil et effectué un nettoyage. Une génération par perturbation (section 6.1.1) pourrait par la suite aider à enrichir les détails des branches.

6.2.4 Usage de notions géométriques

Toutes les opérations décrites jusqu'à présent, qu'elles soient automatiques ou interactives, ne définissent que des méthodes pour générer ou retirer des points, mais rien pour les modifier. Il n'y a pas de reconstruction unique de scène pour les méthodes faisant usage d'un nombre fini d'images en entrée. Sans information additionnelle, le mieux qu'il est possible de faire est de reconstruire l'enveloppe photographique (section 2.1.1). Il est cependant possible d'exploiter les connaissances géométriques de la scène de l'usager. Par exemple, les images des scènes sont typiquement acquises d'objets déposés sur une surface plane. Il est difficile de prendre des images suffisamment au ras du sol (la surface plane) pour bien en saisir la forme. Le résultat est que le plancher reconstruit n'est pas du tout planaire (figure 3.3 (h)). Cependant, l'usager sait très bien qu'il devrait y avoir là un plan. Nous mentionnons là un exemple avec un plan, mais il pourrait tout autant s'agir de formes plus sphériques (fruits, têtes, *etc.*) ou cylindriques (arbres, poteaux, *etc.*) pour n'en nommer que quelques-unes.

Ainsi, nous présentons un outil pour profiter de ces connaissances en effectuant la manipulation de points à l'aide de primitives, un peu comme nous le faisons pour la région d'intérêt (section 6.2.1), mais à la surface de la primitive. L'usager positionne la

primitive quelque part dans la scène en s'aidant des points générés pour la placer. La surface sert alors comme sous-espace d'échantillonnage ou pour déplacer des points vers la surface de la primitive. Nous appelons cette opération une *projection* de points sur la primitive. Cette projection s'effectue en déplaçant le point vers l'endroit le plus près de la surface de la primitive.

Dans l'état actuel de notre système, nous n'avons expérimenté qu'avec le plan comme primitive. Il s'agit du type de surface le plus souvent mal reconstruite dû à la rareté des images qui captent la surface vraiment au ras. De plus, bien que ça exige plus de travail, nous pouvons nous servir de cette primitive pour faire une approximation de plusieurs autres primitives en effectuant des opérations localisées (dessus de table, étagère, *etc.*). L'ajout d'autres types de primitives n'est pas difficile. Il suffit de définir une projection sur la primitive et une méthode d'échantillonnage. Le reste des manipulations est identique à celles du plan, à un détail près. Nous en reparlerons plus loin dans cette section.

Positionnement

Le premier problème auquel il faut s'attaquer est le positionnement (centre, orientation, dimensions) de la primitive dans l'espace. Dans le cas d'un plan, qui est *a priori* infini, nous lui spécifions des dimensions pour des fins d'affichage et de génération, mais nous considérons son infinité pour la projection. Ses dimensions sont celles d'un rectangle qui repose sur le plan. La position et l'orientation du plan donnent le centre et les axes du rectangle.

Considérant que les intentions ici sont de déplacer ou générer de nouveaux points selon les connaissances de l'usager, nous utilisons surtout un positionnement manuel. Il serait possible d'automatiser le positionnement avec des techniques de reconnaissance de formes, mais cela exigerait qu'un bon ensemble de points ait été généré pour que cela fonctionne bien. Nous avons opté pour un positionnement manuel pour plus de flexibilité et il s'agit d'une opération relativement facile pour l'usager.

Il est toutefois possible d'aider l'usager à positionner initialement la primitive soit avec de telles techniques, soit avec des méthodes plus simples comme la méthode de positionnement de la région d'intérêt (section 6.2.1). Nous pouvons également nous servir des points accumulés avec la région d'intérêt pour y ajuster (*fit*) la primitive. Par la suite, l'usager peut effectuer des ajustements manuels au positionnement, si désirés.

Projection de points

Nous pouvons nous servir d'une primitive pour ajuster la position de points. Il nous faut donc un ensemble de points avec lequel travailler. Il y a plusieurs façons de définir cet ensemble. Tout d'abord, nous pouvons nous servir de la région d'intérêt pour une sélection manuelle (section 6.2.1). Une autre méthode serait d'automatiquement sélectionner les points qui se trouvent à une certaine distance (spécifiée par l'usager) de la surface de la primitive.

Une fois que les points sont sélectionnés, nous les déplaçons en les projetant sur la surface de la primitive, *i.e.* nous les déplaçons aux points de la surface le plus près. Cependant, chaque point à sa nouvelle position n'est pas garanti d'être toujours plausible, en particulier si la primitive a été mal placée. Nous effectuons donc un test de plausibilité pour le point à sa nouvelle position et nous effectuons le déplacement seulement s'il est réussi. Il est possible que le test échoue mais qu'à une position légèrement perturbée il soit valide. Nous prenons donc soin d'appliquer une perturbation sur la position et nous pouvons répéter la perturbation plus d'une fois si le test échoue, jusqu'à un nombre maximal d'essais spécifié par l'usager.

Il faut aussi faire attention à la visibilité. Les nouvelles positions pourraient être occultées par les anciennes. Pour la plupart des types de primitives, l'information de visibilité des anciennes positions peut aussi être cruciale, tel qu'illustré à la figure 6.4 pour le cas d'une sphère. Le point P, qui serait projeté à la position P', n'est pas visible des caméras 1 et 2, et pour qu'il soit accepté à la position P', il ne faut toujours pas qu'il soit visible. Par contre, les points plus près des caméras 3 et 4 occultent la position P', alors qu'il ne faudrait pas. Nous avons donc un conflit de visibilité.

Pour remédier à cela, nous pouvons regénérer les cartes de profondeurs en excluant les points sélectionnés, puis effectuer un rendu de la primitive dans chaque carte de profondeurs, sans oublier le biais à ajouter pour éviter d'occulter les positions près de la surface. De cette façon, la position P' de la figure 6.4 serait bel et bien cachée pour les caméras 1 et 2 mais visible pour les caméras 3 et 4.

Le problème ne se pose pas pour une projection sur un plan car aucun point du plan n'en occulte un autre. Il suffit de regénérer les cartes de profondeurs en excluant les points sélectionnés. Il y a bien sûr le cas pathologique d'un point de vue se trouvant sur le plan et regardant dans une direction perpendiculaire à sa normale, mais nous

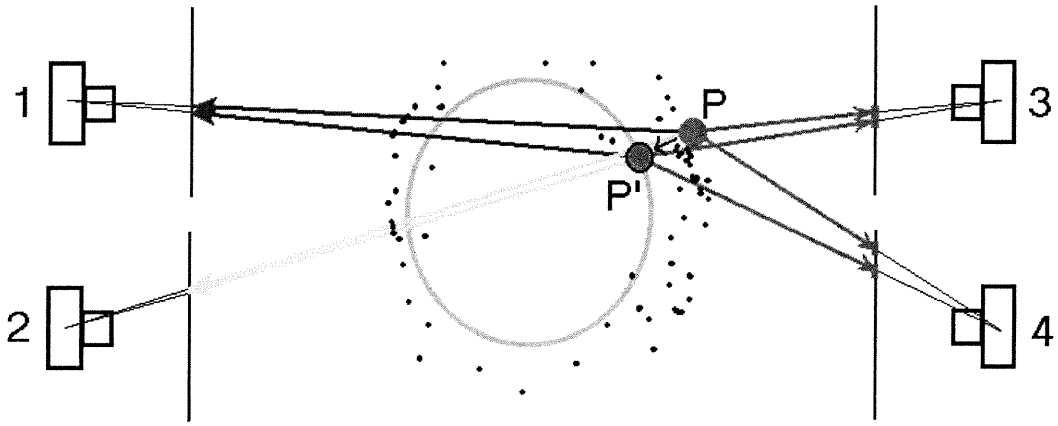


FIG. 6.4 – Problème de visibilité lors de la projection de points sur une surface d'une primitive.

négligeons simplement cette situation bien particulière.

Dans l'exemple du chapitre 3, nous avons employé un plan pour ajuster la position des points qui représentent la surface de la table (figure 3.4 (i) et (j)). La plan a d'abord été positionné en l'ajustant sur les points que l'usager a accumulés avec la région d'intérêt. Puis les points ont été projetés sur le plan.

Génération de points

Nous pouvons également nous servir de la primitive pour restreindre la génération de points sur sa surface. Ceci est pratique pour enrichir les détails de la région après projection sur la primitive. Pour générer des points, il nous faut un moyen d'échantillonner la surface de la primitive. Dans le cas d'un plan, il suffit d'interpoler linéairement les quatre sommets de son rectangle.

Lorsqu'un point est généré, il faut s'assurer qu'il soit à l'intérieur du volume de reconstruction, car rien n'empêche que la primitive n'y soit pas entièrement incluse. Un test pourrait être effectué pour vérifier cette inclusion et éviterait alors d'effectuer le test pour chaque point.

À la figure 3.4 (k), nous nous sommes servi du plan pour combler les trous de la surface après projection des points. Les dimensions du plan ont été réduites par rapport au plan de l'étape précédente en (j) pour éviter de générer trop de positions en dehors du volume de reconstruction.

Nous pouvons aussi nous servir de la génération sur un plan pour incorporer une méthode de reconstruction : la coloration de voxels [SD99], pourvu que la scène à re-

construire se situe à l'extérieur de l'enveloppe convexe formée par les points de vue. L'usager n'a alors qu'à placer le plan de sorte qu'il soit près de la frontière du volume de reconstruction et de l'ensemble des points. Ses dimensions doivent aussi couvrir tout le volume. Ensuite, il suffit de générer un ensemble de points sur le plan (la quantité maximale est définie par l'usager) puis de générer les cartes de profondeurs. Ensuite, nous déplaçons le plan légèrement le long de sa normale en nous éloignant des points de vue et nous recommençons. Nous effectuons ces itérations de façon automatique jusqu'à ce que le plan sorte au complet du volume de reconstruction. Pour être fidèle à la coloration de voxels, le plan est échantillonné régulièrement, mais il serait possible de le faire plus aléatoirement avec une densité suffisante de points. Étant donné que nous ne colorons pas vraiment des voxels, nous appelons cette méthode le *balayage de plan*. La figure 6.5 montre deux exemples de résultats obtenus à l'aide du balayage de plan.

(a) Balayage de plan pour la scène des cordes d'escalades (section 8.1)

(b) Balayage de plan pour la scène du buisson synthétique (section 8.2)

FIG. 6.5 – Exemples de résultats pour le balayage de plan.

Il serait possible d'effectuer une génération similaire à la sculpture d'espace [KS00], seulement l'approche serait incrémentielle et l'implantation ne serait pas aussi directe que pour la coloration de voxels décrite ci-avant. Il s'agirait essentiellement de procéder comme pour cette dernière, sauf que pour un positionnement donné d'un plan, il faudrait considérer seulement les points de vue qui se trouvent du côté opposé au déplacement du plan. Pour les autres passes, il faudrait ajouter une étape de nettoyage après chaque génération de points et de cartes de profondeurs. Pour ce nettoyage, il faudrait considérer seulement les points de vue et les points qui se trouvent du côté opposé au déplacement

du plan. Ce ne serait pas vraiment pas une implantation efficace de la sculpture d'espace (nous ne profitons pas de la cohérence d'une grille tridimensionnelle), mais l'idée est de montrer qu'il est possible de l'incorporer à notre système. Et de fait que nous ne sommes pas contraints à une grille fixe, nous pouvons effectuer des passes avec des angles autres que perpendiculaires entre eux.

Chapitre 7

Affichage

Nous présentons dans ce chapitre les techniques employées pour l'affichage de la scène reconstruite. Nous avons deux buts principaux, et souvent contradictoires, à atteindre :

- l'interactivité pour le feed-back lors de la séance de modélisation ;
- la qualité du rendu pour la reconstruction finale.

À la section 7.1, nous présentons une méthode d'affichage employée pour maximiser l'interactivité. Compte tenu de l'importance de l'interactivité de notre technique, elle sera la plus utilisée. À la section 7.2, nous présentons une méthode visant d'abord un rendu de meilleure qualité, que nous pouvons utiliser pour afficher les détails qui dépendent plus de l'angle de vue de la scène. Cette technique sera plus propice pour afficher les effets spéculaires. Enfin, nous présentons certaines améliorations et d'autres techniques applicables à la section 7.3.

7.1 Rendu OpenGL

Lors d'une séance de reconstruction interactive, il est important que l'affichage du résultat courant soit le plus rapide possible afin de donner un feed-back immédiat à l'usager selon les opérations qu'il applique à la scène à reconstruire. Nous effectuons cet affichage par l'intermédiaire de la simple primitive `GL_POINTS` d'OpenGL (chapitre 2 de [WNDS97]). Nous profitons également des tableaux de points (*vertex arrays*, même chapitre) pour accélérer le rendu de l'ensemble des points, d'autant plus que nos points sont déjà emmagasinés dans un tableau. Sur une machine équipée d'une carte vidéo *GeForce3* de *n*VIDIA [*n*VIDIA], nous obtenons des taux de rafraîchissement entre 20 et 30 images par seconde pour plusieurs centaines de milliers de points, ce qui est amplement suffisant pour effectuer un rendu d'un objet complexe en temps réel.

L'usager doit spécifier la taille d'affichage des points en pixel. Cette taille fixe est

la dimension d'un carré centré à la position bidimensionnelle où un point est projeté. Typiquement, au début de la séance de modélisation, l'usager spécifie une taille plus grande afin de mieux combler les trous pour la visualisation d'un petit ensemble de points et d'aider la sélection de points avec la souris. En pratique, une taille de quatre par quatre pixels s'est avérée un bon choix. Plus l'ensemble de points s'enrichit, moins il y a de trous et, pour mieux visualiser les détails, dépendamment de la proximité du point de vue de la scène, il faut réduire cette taille. La taille minimale est de un pixel.

De façon similaire à la taille d'affichage, l'usager doit également spécifier une taille de rendu des points pour la génération des cartes de profondeurs. Typiquement nous prenons toujours une taille de un pixel. Pour une taille plus grande, chaque point couvre une plus grande surface et peut cacher ses voisins. Ceci n'est pas souhaitable pour un ensemble de points dense, mais par contre au début du processus de reconstruction, ceci peut contribuer à limiter la génération de points faux.

À moins d'avis contraire, l'ensemble des exemples présentés dans ce mémoire ont été rendus avec cette méthode.

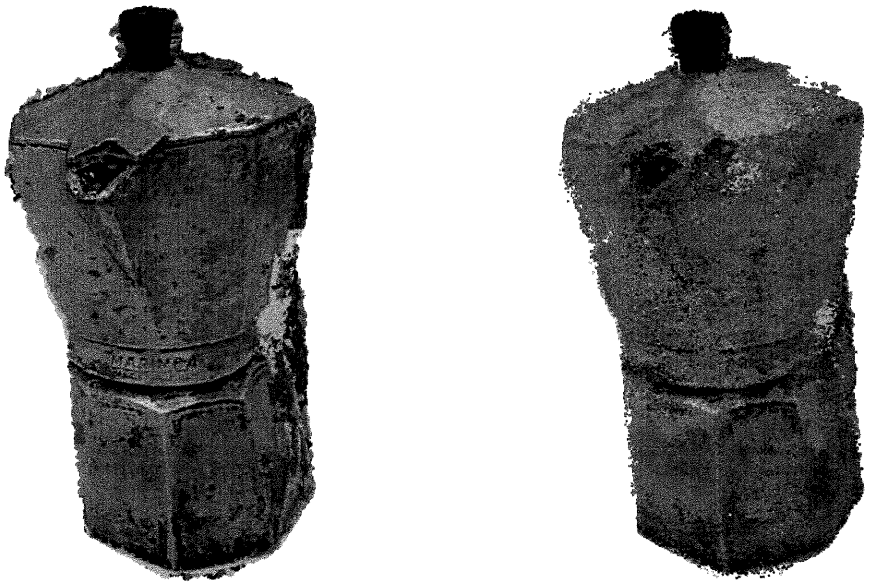
7.2 Reprojection de textures

Pour obtenir un affichage de plus grande qualité, nous avons exploité quelques idées provenant du lumigraphe non structuré (*unstructured lumigraph*) [BBM+01], et aussi des textures de [DTM96]. Pour ce faire, nous effectuons un premier rendu avec la méthode décrite à la section précédente, puis nous consultons le contenu de la carte de profondeurs produite. Pour chaque pixel P, nous retrouvons sa position tridimensionnelle p en multipliant les coordonnées de P avec sa profondeur par l'inverse de la matrice de projection pour la vue courante. Nous retrouvons ensuite le point de vue dans les images originales, qui voit p avec l'angle le plus similaire au point de vue courant. Nous projetons ensuite p dans la caméra correspondante, lisons la couleur (originale, non quantifiée) du pixel et nous attribuons cette couleur à la position de P pour la vue courante. Il s'agit d'une version simplifiée de la technique présentée par Buehler *et al.* [BBM+01], n'utilisant que le critère d'angle pour le choix de la caméra. Ce critère est réévalué à chaque pixel de la vue courante où un point projette.

Il est à noter que la taille en pixel d'affichage des points n'a pas la même influence que celle du rendu OpenGL. Au contraire, une taille plus grande donne généralement de meilleurs résultats. En effet, puisque la couleur finale d'un pixel n'est pas donnée par

la couleur du point mais par la reprojection de ses pixels, l'impact de la superposition de points voisins est amoindrie.

Il serait possible d'optimiser cette approche, notamment à l'aide des récents développements au niveau du matériel graphique (pipeline programmable [OHHM02]). Cependant, avec notre implantation courante, nous obtenons déjà des taux de rafraîchissement d'environ deux images par seconde, dépendamment de la résolution d'affichage.



(a) Rendu avec reprojection de textures. (b) Rendu OpenGL du même point de vue.

FIG. 7.1 – Rendu par reprojection de textures *vs.* rendu OpenGL.

La figure 7.1 (a) illustre un exemple de rendu avec notre méthode de reprojection de textures pour la scène de la cafetière (section 8.3). Nous pouvons comparer avec le résultat de la figure 7.1 (b). Dans le premier cas, les détails spéculaires et du texte gravé apparaissent bien, alors qu'ils sont pratiquement inexistants pour le rendu OpenGL.

Bien que cette technique produise en général des images de meilleure qualité, un problème peut survenir lors de la reprojection : des "fantômes" d'objets hors du volume de reconstruction peuvent apparaître, soit, dans une image, un objet se trouvant hors du volume de reconstruction mais obstruant au moins partiellement la scène à reconstruire. Dès que nous regardons la scène à partir d'un point de vue près de celui de cette image, l'objet obstruant apparaîtra comme s'il était projeté sur les points de la scène. La

figure 7.2 illustre un exemple d'une telle situation avec la scène des cordes d'escalade du chapitre 3. Pour réduire ces fantômes, il faut faire attention à la prise des images et aux occultations causées par des objets hors de la région de la scène à reconstruire.

(a) Rendu OpenGL

(b) Rendu par reprojection de textures. Nous pouvons distinguer le "fantôme" d'un pot de colle. Celui d'un crayon est moins visible.

(c) L'image globalement la plus près du point de vue.

FIG. 7.2 – "Fantômes" de la reprojection de textures.

Enfin, nous rappelons que l'un des buts du rendu par points est d'éviter les grands coûts en mémoire des méthodes de rendu par images, et donc notre rendu par reprojection peut paraître contradictoire. Cependant, il est question ici de seulement 10 ou 20 images (typiquement) au lieu des quelques centaines des méthodes de rendu par images, ce qui est raisonnable pour une station de travail actuelle. Un exemple présenté par Buehler *et al.* [BBM+01] n'a que 36 images, mais, tout comme nous, ils font usage d'un modèle géométrique comme mandataire (*proxy*) qui représente déjà bien la forme de l'objet (une voiture).

7.3 Améliorations et autres techniques

Il est possible d'améliorer le rendu par reprojection de textures en effectuant un filtrage de la carte de profondeurs. Les filtres lissants (*blur*), tel un filtre gaussien, peuvent réduire les artefacts causés par le bruit des points de la scène reconstruite. Des techniques de remplissage de trous peuvent réduire le sous-échantillonnage dans la représentation par points. Un exemple de ces techniques est de tout simplement effectuer une moyenne des profondeurs des pixels dans un voisinage, pourvu que ce voisinage ne soit pas trop vide. Dépendamment du type d'opération et de la taille des filtres, tout ceci réduit d'avantage le taux de rafraîchissement. Il serait possible cependant de profiter des récents développements en matériel graphique pour accélérer le tout à l'aide des

pipelines programmables. Ce type de filtrage peut nuire cependant dans les zones ayant beaucoup de discontinuités, lissant des zones qui devraient apparaître bruitées.

Enfin, comme notre méthode de reconstruction produit une représentation par points, la plupart des méthodes actuelles de rendu par points (notamment les *surfels* [PZvBG00] et le *splatting* de surfaces [ZPvBG01]) pourraient servir à produire des rendus de qualité. Cependant, ces méthodes exigent une normale à chaque point, ce que notre système *a priori* ne calcule pas. Il serait cependant possible d'extraire des normales d'une façon similaire la méthode de Linsen [Lin01], *i.e.* en prenant la normale d'un plan ajusté au point concerné dans un certain voisinage. Ces normales pourraient être faussées par le bruit des points ou par une grande quantité de discontinuités.

Chapitre 8

Résultats

Nous présentons dans ce chapitre des détails statistiques pour l'ensemble des exemples retrouvés dans les chapitres précédents ainsi que quelques résultats pour des scènes supplémentaires. Les résultats sont présentés par scène : les cordes d'escalade (section 8.1), le buisson synthétique (section 8.2), la cafetière (section 8.3), le sandwich (section 8.4) et la statuette (section 8.5).

Les images de scènes réelles ont été prises avec une caméra numérique de modèle soit *Canon EOS-DS30* ou alors *Kodak DC290*. Les tests ont été effectués sur une machine avec deux UCTs (*CPU*s) de type *AMD Athlon* 1.2 Ghz (mais une seule UCT a été utilisée) avec 1 Go de mémoire et une carte vidéo *GeForce3* de *n*VIDIA, sauf les exemples de la section 8.1 et la méthode interactive de la section 8.3, où une machine avec une UCT de type *AMD Athlon XP 2100* 1.7 Ghz avec 512 Mo de mémoire et une carte vidéo *GeForce4* de *n*VIDIA a été employée. Aucune optimisation n'a été utilisée lors de la compilation pour des fins de débogage lors du développement. Typiquement, avec le compilateur employé (*gcc*), l'optimisation maximale est de l'ordre de 2 à 2.5 fois plus rapide au niveau du temps d'exécution.

Dans tous nos tests, le seuil pour le test de plausibilité est de 80%, *i.e.* la couleur dominante doit se retrouver dans au moins huit images sur dix visibles. Le nombre d'images minimum dans lesquelles un point doit être visible est deux.

8.1 Cordes d'escalade

Notre principale scène pour nos tests est un monceau de cordes d'escalade. L'ensemble des images de la scène ainsi que le volume de reconstruction sont présentés à la figure 8.1. Il s'agit d'une scène quasi-idéale au niveau de la réflectance des surfaces. En effet, les cordes sont très diffuses. Il n'y a que les mousquetons et la plaque frein qui

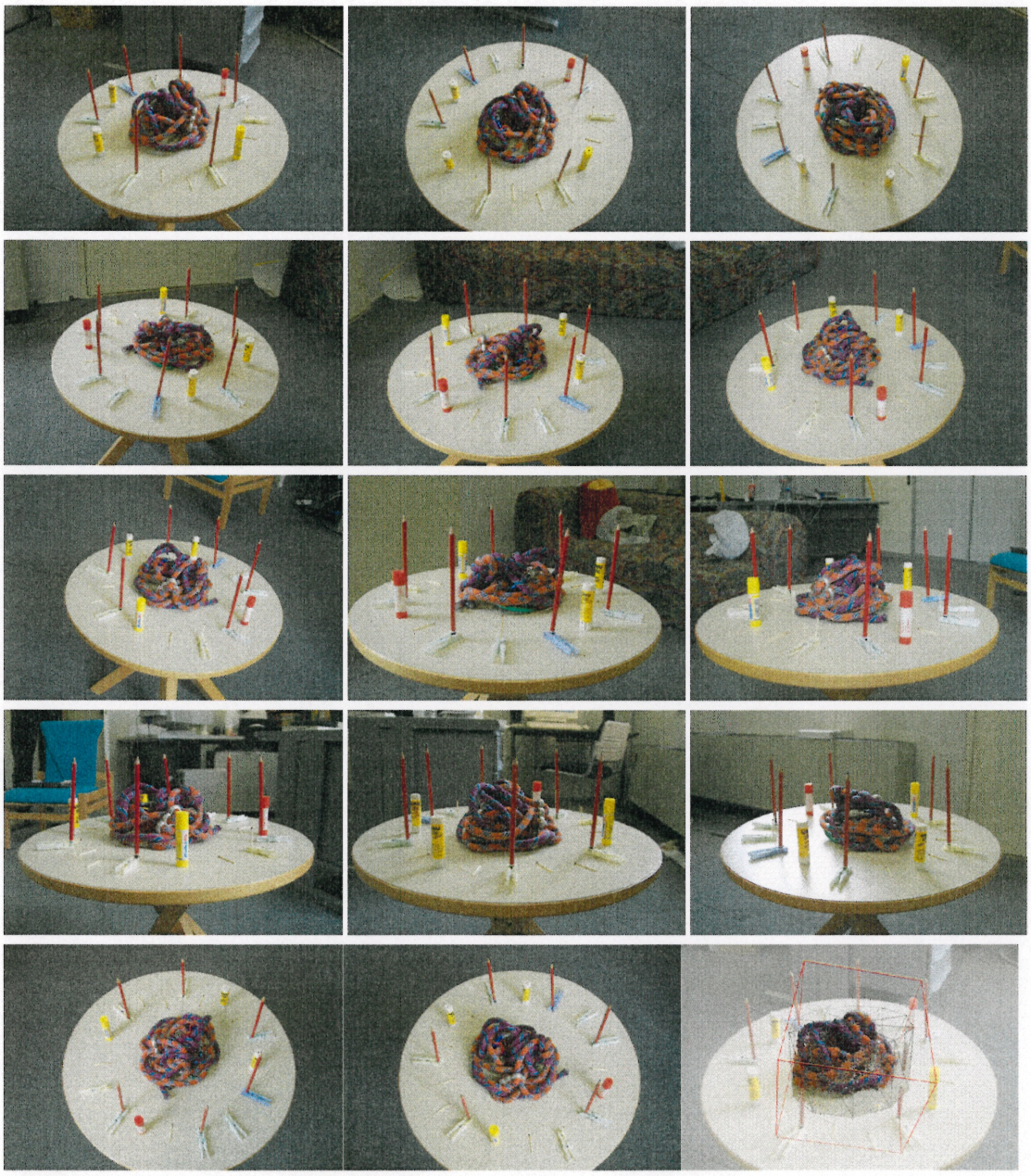


FIG. 8.1 – Les 14 images de la scène des cordes d'escalade et le volume de reconstruction en bas à droite. Résolution des images : 2 160 × 1 440 pixels.

sont plus spéculaires. La texture des cordes présente des détails fins, des changements de couleurs brusques, et il y a typiquement de bons écarts de visibilité d'une image à l'autre même si elles sont prises de points de vue similaires.

Cette scène a été utilisée comme séance typique de modélisation au chapitre 3. Nous présentons plus de détails statistiques de la séance à la section 8.1.1. Bien que nous ayons essentiellement fait une quantification à 10 couleurs pour cette scène, notamment pour la séance typique, nous présentons aussi quelques résultats pour des quantifications à des nombres différents de couleurs ainsi que pour d'autres types de comparaisons de couleurs à la section 8.1.2. Enfin, nous présentons quelques résultats sur l'usage de la perturbation des points à la section 8.1.3.

8.1.1 Séance typique de modélisation

Étape	Nombre de points	Nombre d'itérations	Temps (s)	Nombre de tests	Taux d'acceptations	Visibilité moyenne
(a)	26 638	22	238	2 430 952	10.6%	89.3%
(b)	103 809	12	170	2 186 861	43.9%	58.5%
(c)	..... *positionnement de la sphère*					
(d)	144 282	14	113	1 365 275	10.6%	68.2%
(e)	126 141	—	7	—	87%	—
(f)	..... *zoom*					
(g)	168 131	100	469	10 273 300	8.4%	84.8%
(h)	221 420	126	1 083	11 697 407	47.2%	50.6%
(i)	..... *positionnement du plan*					
(j)	221 420	100	30	1 485 503	3.4%	73.8%
(k)	358 421	6	114	2 168 610	87%	38.2%
(l)	200 968	83	60	—	56%	—

TAB. 8.1 – Statistiques pour les étapes des figures 3.2, 3.3 et 3.4 de la séance typique de modélisation du chapitre 3.

Le tableau 8.1 donne des statistiques sur les diverses étapes, illustrées aux figures 3.2, 3.3 et 3.4, de la séance typique de modélisation du chapitre 3. Le **nombre de points** indique combien de points présents à la fin des opérations de l'étape. Le **nombre d'itérations** correspond soit au nombre de boucles effectuées dans le processus de génération, soit au nombre d'opérations effectuées avec un outil, ou alors au nombre maximal d'essais permis pour générer ou déplacer un point. Le **temps** (en secondes) est le temps pris par l'exécution de l'étape. Le **nombre de tests** correspond au nombre de fois que le test de plausibilité a été effectué. Le **taux d'acceptations** est soit la proportion des tests de plausibilité ayant réussi, soit la proportion des points conservés

après fusion. Enfin, la **visibilité moyenne** correspond à la moyenne des ratios entre le nombre d'images dans lesquelles un point est visible et le nombre total d'images lors du test de plausibilité. Ci-après nous présentons une description plus détaillée de chaque étape. Il est à noter que la comparaison des couleurs a été effectuée par l'intermédiaire de la quantification (10 couleurs).

(a) Pour générer l'ensemble de points initiaux, nous avons appliqué le processus de génération de base où chaque itération consistait à tester 100 000 positions à l'intérieur du volume de reconstruction, puis à mettre à jour les cartes de profondeurs et enfin à effectuer un nettoyage. La visiblité moyenne est élevée étant donné qu'il y a peu de points, ce qui fait en sorte que le taux d'acceptations est faible car nous avons encore peu d'information sur la visibilité.

(b) À partir du moment où nous avons plus d'information sur la visibilité et du moment où, à partir de l'ensemble initial de points, nous percevons mieux la scène reconstruite, nous pouvons profiter de la distribution actuelle des points pour accélérer la génération. À cette étape, nous avons employé le même processus qu'à l'étape (a), mais juste avant la mise à jour des cartes de profondeurs nous effectuons une génération par perturbation (section 6.1.1) à partir des points qui ont été générés parmi les dernières 100 000 positions testées. En seulement 12 itérations, l'ensemble de points s'est beaucoup raffiné (un gain de presque 300%). Bien entendu, le taux d'acceptations est beaucoup plus élevé qu'à l'étape (a) car les points produits par la perturbation ont en général une forte probabilité de réussir le test de plausibilité. Avec plus de points, les cartes de profondeurs donnent plus d'information sur la visibilité et donc la visibilité moyenne est réduite.

(c) L'identification de régions d'intérêt s'effectue rapidement à l'aide de notre méthode de positionnement (section 6.2.1). La nature interactive de ce dernier permet d'employer la région d'intérêt comme un pinceau d'opérations appliquées en trois dimensions là où elle se situe.

(d) La région d'intérêt de l'étape (c) est déplacée à 14 endroits différents. À l'intérieur de celle-ci, le même processus qu'à l'étape (b) est appliqué. Le temps d'exécution indiqué au tableau 8.1 inclus le temps pris pour changer le point de vue et pour positionner la région d'intérêt (ce temps est pratiquement négligeable). Le taux d'acceptations a chuté considérablement. Ceci provient du fait que, comme nous

générons des points dans une sphère dont le centre se situe à peu près sur la surface de l'objet, une bonne proportion des points sont générés à des positions autres qu'à la surface (à l'intérieur ou à l'extérieur de l'objet) où ils sont rejetés. Par contre, étant donné que nous testons beaucoup (100 000) de positions à l'intérieur de la sphère, les points sur la surface ressortent assez rapidement. Bien que le taux d'acceptations soit faible, cette méthode demeure généralement plus efficace que la méthode de génération de base dans le volume de reconstruction au complet, car seulement les points inclus dans la sphère sont considérés pour le nettoyage effectué à la fin de chaque opération avec la région d'intérêt.

(e) Une fusion (section 6.1.2) est effectuée pour réduire les redondances. Une grille de $293 \times 224 \times 256$ voxels est employée, ce qui est relativement fin, mais nécessaire pour ne pas détériorer la qualité de l'ensemble de points. La figure 8.2 compare le résultat de la fusion de l'ensemble de points obtenu à l'étape (d) avec des grilles de deux différentes résolutions. Avec la plus grande résolution, seulement 13% des points sont éliminés, ce qui indique un faible taux de redondance. Les 7 secondes requises pour cette simple opération sont élevées dû aux fautes de pages (*page faults*, ou *swap*), *i.e.* nous avions rempli la mémoire de la machine. Par contre, la même opération à l'étape (l) n'a pris que 2.7 secondes pour une même résolution et plus de points (le *swap* avait déjà été fait).

(a) Ensemble de 144 282 points.

(b) Après fusion de (a) avec une grille de $293 \times 224 \times 256$ voxels. L'ensemble contient maintenant 126 141 points.

(c) Après fusion de (a) avec une grille de $115 \times 88 \times 100$ voxels. L'ensemble contient maintenant 70 388 points.

FIG. 8.2 – Comparaison de fusion avec deux grilles de dimensions différentes.

(f) Un simple zoom est effectué.

(g) Le remplissage de la vue (section 6.2.2) est effectué avec 100 essais par trou et en utilisant la génération à proximité du point de vue. Des 102 733 trous identifiés,

41 990 ont été comblés, ce qui paraît peut (moins de 50%), mais plusieurs de ces trous se trouvent hors de la scène, d'où également le faible taux d'acceptations. La faible valeur de ce dernier est aussi due aux multiples essais par trou.

(h) À cette étape, la génération de points s'est poursuivie avec les méthodes des étapes (b) et surtout (d). Quelques fusions ont également été effectuées.

(i) Un plan est positionné pour corriger les points (section 6.2.4) issus de la surface de la table sur laquelle reposent les cordes. Les points à projeter ont été sélectionnés à l'aide de la région d'intérêt ; le plan a d'abord été ajusté par moindres carrés à ces points, puis légèrement ajusté manuellement. Le calcul d'ajustement du plan s'effectue en moins d'une seconde, mais la sélection des points requiert plus de travail de la part de l'usager (de l'ordre d'une à deux minutes). Un total de 53 953 points ont été sélectionnés.

(j) Plus d'une projection des points sont effectuées. À la première projection, 45 381 points sont déplacés sur le plan, avec un maximum de 100 essais avec perturbation. Les points restants se trouvent typiquement en bordure du volume de reconstruction. D'autres projections pour ces points sont effectuées. Cependant, si le plan n'est pas perpendiculaire au volume, comme c'est le cas ici, des points peuvent être projetés sur le plan à l'extérieur du volume, et si la perturbation n'est pas assez grande, leur projection ne sera jamais acceptée, ce qui explique, avec les essais multiples, le faible taux d'acceptions.

(k) Plus de points sont générés sur le plan. Les dimensions de ce dernier ont été réduites pour limiter la quantité de positions générées à l'extérieur du volume de reconstruction. La méthode de génération est la même qu'à l'étape (b), ce qui explique le haut taux d'acceptations malgré la faible visibilité moyenne. En effet, un échantillonnage uniforme du plan produit une grande quantité de points rejetés parce qu'ils sont sous les cordes. Cependant la plupart des points générés à côté des cordes sont acceptés et, par la génération par perturbation, ils sont presque doublés.

(l) La fusion à la fin emploie une grille de même résolution qu'à l'étape (e), mais n'a pris que 2.7 secondes. Le reste des 60 secondes est passé à supprimer des points que l'usager estime être à des positions aberrantes. La suppression de 543 points a été effectuée à l'aide de la région d'intérêt à 83 positions.

En incluant le temps requis pour les prises de décisions et les manipulations de l'interface du système, le temps total de la séance est d'environ 45 minutes. Ceci peut paraître long comparativement aux 10.5 minutes requises pour produire le résultat de la figure 6.5 (a). Ce dernier ensemble de 517 307 points a été généré avec un balayage de plan (section 6.2.4) en 286 itérations à 100 334 échantillons par itération. Cependant, par chance, cette scène se trouve à l'extérieur de l'enveloppe convexe des points de vue des images (condition nécessaire pour la coloration de voxel, section 2.1.1). Ensuite, il faudrait y ajouter les manipulations pour corriger les points représentant la table sur laquelle reposent les cordes. Mais il est clair que si cette contrainte est satisfaite, le balayage de plan est plus efficace.

8.1.2 Comparaison de couleurs

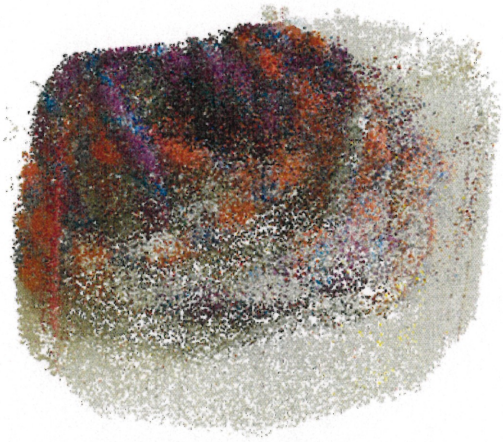
Le niveau de la quantification et le type de comparaison de couleurs affectent la qualité et l'efficacité de la reconstruction. La figure 8.3 illustre le résultat de diverses méthodes de comparaison de couleurs. Nous avons employé seulement la méthode de génération de base pour produire les résultats de la figure 8.3. À chaque itération, 500 000 points ont été testés. Le tableau 8.2 montre quelques statistiques sur la production de ces résultats.

Méthode	Nombre de points	Nombre d'itérations	Temps (s)
Quantification (5 couleurs)	158 047		90
Quantification (10 couleurs)	305 393	20	1 538
Quantification (30 couleurs)	310 712	63	4 587
Distance (RGB)	313 871	32	8 614

TAB. 8.2 – Statistiques pour diverses méthodes de comparaison de couleurs.

Globalement, la quantification produit des résultats beaucoup plus rapidement qu'avec la comparaison par distance entre couleurs. Cependant, cette dernière tend à produire des résultats plus précis au niveau de la texture. En effet, étant donné qu'avec la quantification nous perdons nécessairement certains détails dans les images, il est certain que la comparaison à partir des couleurs des images d'origine permettra de mieux les cerner.

Au niveau de la quantification, moins il y a de couleurs, plus la production est rapide. Cependant, moins il y a de couleurs, moins le résultat est précis. Des exemples d'images produites pour la quantification à 5, 10 et 30 couleurs se trouvent à la figure 5.1



(a) Quantification à 5 couleurs.

(b) Quantification à 10 couleurs.

(c) Quantification à 30 couleurs.

(d) Distance Euclidienne entre les couleurs dans l'espace RGB.

FIG. 8.3 – Effets sur la reconstruction de la quantification et de la comparaison par distance entre couleurs.

(section 5.2.2). La figure 8.3 (a) illustre bien les insuffisances de 5 couleurs. Par contre, entre 10 et 30 couleurs, la différence de qualité n'est pas très grande, mais le temps d'exécution est considérablement plus long avec 30 couleurs. Ceci illustre l'importance du choix du nombre de couleurs. Pour avoir le meilleur rapport entre la qualité et le temps d'exécution, il faut choisir le nombre minimal de couleurs de sorte que les détails ressortent encore suffisamment dans les images quantifiées.

8.1.3 Perturbation

(a) Avant la perturbation. (b) Après la perturbation. (c) Différence absolue des images (en couleurs inversées).

FIG. 8.4 – Perturbation sur un ensemble de 97 169 points.

(a) Avant la perturbation. (b) Après la perturbation. (c) Différence absolue des images (en couleurs inversées).

FIG. 8.5 – Perturbation sur un ensemble de 369 868 points.

Nous avons présenté la perturbation de points (section 6.1.1) comme une méthode pour améliorer la qualité de l'ensemble actuel de points. Or, il s'est avéré qu'en pratique, l'impact de cette opération n'est pas aussi grand que nous l'aurions espéré. Les figures 8.4 et 8.5 illustrent ceci. Nous avons effectué un test pour deux ensembles de points de taille différente. L'un a 97 169 points et le résultat est encore grossier ; l'autre a 369 868

points et le résultat est beaucoup plus raffiné.

Visuellement, nous observons peu de différences entre avant l'application de la perturbation et après, en particulier pour l'ensemble de points plus raffiné, d'autant plus que nous avons exécuté la perturbation pendant presque 15 heures. Les images de droite de chaque figure montrent la différence en valeur absolue (en couleurs inversées) entre les images avant et après perturbation pour aider à mieux visualiser les différences des résultats. Dans le cas de l'ensemble de points plus grossier, la perturbation a été exécutée pendant un peu moins de 20 minutes. La différence est un peu plus grande, mais le temps d'exécution demeure considérable pour le peu de gain en qualité. Et encore, visuellement, l'ensemble de points avant la perturbation est plus plaisant qu'après, même si la silhouette des cordes est un peu mieux définie après.

8.2 Buisson synthétique

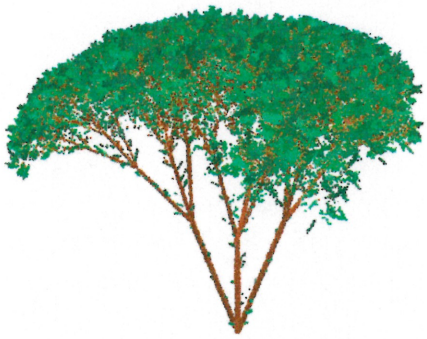
La scène du buisson synthétique est très complexe en terme d'occultations entre les feuilles. Elle comporte également plusieurs structures fines au niveau des branches. La figure 8.6 montre l'ensemble des images de la scène. Le buisson a été généré à l'aide d'un système-L simple provenant de Prusinkiewicz et Lindenmayer [PL90] et les images ont été produites avec un rendu OpenGL où l'illumination et la réflectance étaient diffuses. Un anti-aliassage par sur-échantillonnage et perturbation a également été appliqué. Aucune étape de calibration n'était nécessaire car nous connaissions tous les paramètres des caméras (synthétiques) pour chaque rendu.

Étant donné que le fond de chaque image est blanc, nous avons fait usage de masques de rejet produits automatiquement en effectuant une simple substitution de couleurs (un exemple se trouve à la section 5.2.5) : tout ce qui n'est pas blanc est remplacé par la couleur noir.

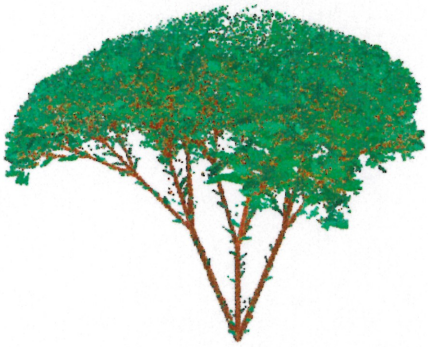
La figure 8.7 montre divers résultats obtenus pour cette scène. La comparaison des couleurs est effectuée par l'intermédiaire de la quantification (15 couleurs). À la figure 8.7 (a), nous avons le résultat obtenu pour le processus de génération de base. Les statistiques du tableau 8.3 (première ligne) ne sont pas très encourageantes, d'autant plus qu'après fusion dans une grille dense de $339 \times 262 \times 305$ voxels, il ne reste plus que 11 583 points, ce qui veut dire que la méthode de reconstruction a eu beaucoup de difficultés pour produire des points dans des régions encore vides. Même avec l'aide de la région d'intérêt, les régions vides ne se remplissent pas.

FIG. 8.6 – Les 20 images de la scène du buisson synthétique et le volume de reconstruction en bas. Résolution des images : 1 420 × 1 085 pixels.

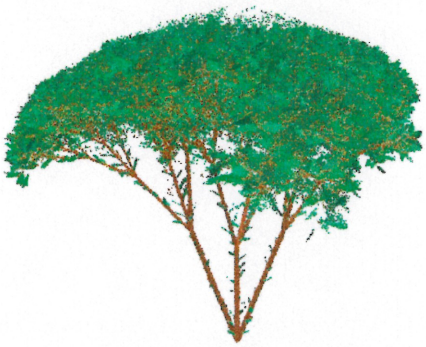
(a) Génération avec toutes les images.



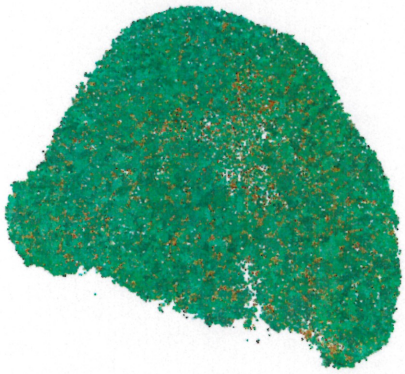
(b) Génération avec les images du dessus.



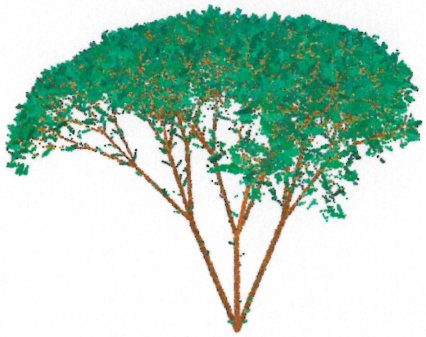
(c) Génération avec les images du dessous.



(d) Combinaison de (b) et (c).



(e) Combinaison de (b) et (c), vue du haut : les ambiguïtés de (c) ressortent.



(f) Après une étape de nettoyage des points de (e).

FIG. 8.7 – Résultats obtenus pour la scène du buisson synthétique.

Cas	Nombre de points testés par itération	Nombre de points	Nombre d'itérations	Temps	Nombre de points après fusion
Toutes les images	500 000	312 420	8 439	4 jours	11 583
Images du dessus	100 000	500 204	2 218	11.5 h	223 393
Images du dessous	500 000	500 810	557	3.2 h	196 468

TAB. 8.3 – Statistiques pour la reconstruction de la scène du buisson.

Par contre, si nous exécutons le même processus mais seulement avec les images qui regardent le buisson du dessus (ou du dessous), nous obtenons respectivement les résultats des figures 8.7 (b) et (c). Comme l'indique le tableau 8.3, les temps d'exécution demeurent longs, mais nous obtenons de bien meilleurs résultats en beaucoup moins de temps. L'un des problèmes avec cette scène est que dans le modèle géométrique initial du buisson les feuilles ont une épaisseur nulle, et elles n'ont pas la même couleur dessus que dessous. Ceci crée des problèmes au niveau des cartes de profondeurs lorsque le biais est ajouté. Mais si nous considérons seulement les images qui ne voient qu'une seule face des feuilles, alors nous pouvons reconstruire beaucoup mieux le feuillage.

Un autre problème provient de la forme du volume de reconstruction. Beaucoup d'espace vide dans le volume rend inefficace un échantillonnage uniforme, ce qui explique les longs temps d'exécution pour les sous-ensembles d'images. La méthode de génération par balayage de plan (section 6.2.4) est plus efficace. Pour obtenir le résultat de la figure 6.5 (b), 246 itérations de générations sur un plan ont été faites. Le plan était échantillonné avec 100 868 points. Un total de 107 851 points ont été produits en environ 18 minutes.

Pour obtenir une reconstruction plus complète, nous pouvons combiner les résultats produits par le sous-ensemble des images du dessous et celui des images du dessus, tel que présenté à la figure 8.7 (d). Cependant, pour chaque sous-ensemble, par la propriété de l'enveloppe photographique, plusieurs points sont faux si nous considérons l'ensemble entier des images. La figure 8.7 (e) met ceci en évidence en montrant un point de vue différent du résultat de la combinaison. Tel que visible dans la dernière rangée d'images de la figure 8.6, le centre du buisson devrait être vide. Cependant les images du dessous n'ont pas de points de vue mettant ce vide en évidence comme dans les images du dessus. La combinaison des deux sous-ensembles nous donne beaucoup d'information sur la visibilité ; nous pouvons alors effectuer un nettoyage dont le résultat est présenté à la figure 8.7 (f). Après cette opération, nous avons tout de même perdu plus de détails que nous l'aurions souhaité, mais il ne faut pas oublier que le problème d'épaisseur nulle des feuilles persiste.

Nous glissons un dernier mot sur la différence entre le temps d'exécution pour le sous-ensemble des images du dessus et celui des images du dessous. Le premier sous-ensemble a plus d'images que l'autre, le temps d'exécution devrait donc être plus élevé,

mais pas à ce point. La différence réside dans le nombre de points testés à chaque itération. Étant donné qu'à la fin de chaque itération une opération de nettoyage est effectuée sur l'ensemble des points, tester moins de points par itération fait en sorte d'augmenter considérablement le nombre de fois que chaque point est revalidé.

8.3 Cafetière

La scène de la cafetière est *a priori* peu complexe. La cafetière, dont les images sont présentées à la figure 8.8, est un objet dont la surface est lisse et qui pose peu de difficultés au niveau de la visibilité (si ce n'est de l'anse). Cependant, sa surface est luisante, ce qui peut poser des difficultés pour les mises en correspondance. Nous avons fait usage de masques de rejet pour cette scène car le volume de reconstruction est beaucoup plus grand qu'il ne le faut, dû à des problèmes rencontrés lors de la calibration. Ceci a permis d'accélérer la génération de points en écartant rapidement ceux ne se trouvant pas dans la cafetière.

Méthode	Nombre de points	Nombre d'itérations	Temps
De base avec quantification (10 couleurs)	500 256	921	6.2 h
De base avec distance (RGB)	504 734	178	4 h
Interactive avec quantification (10 couleurs)	506 979	570	1.2 h

TAB. 8.4 – Statistiques pour la reconstruction de la scène de la cafetière.

Malgré la surface luisante de la cafetière, nous arrivons à obtenir des résultats intéressants. Nous avons employé trois méthodes de reconstruction, dont les statistiques sont présentées au tableau 8.4.

Méthode de base avec quantification : Nous avons employé la méthode itérative de base avec 100 000 points testés par itération. La comparaison a été effectuée par l'intermédiaire de la quantification (10 couleurs). Tout comme il a été observé pour la scène du buisson synthétique (section 8.2), le temps d'exécution aurait été beaucoup moindre (entre deux et trois heures) si nous avions testé plus de 100 000 points (500 000 par exemple) par itération. Le résultat est affiché à la figure 8.9 (a).

Méthode de base avec comparaison par distance entre couleurs : Il s'agit de la même méthode que la précédente, sauf que la comparaison des couleurs est

FIG. 8.8 – Les neuf images de la scène de la cafetière avec leurs masques de rejet correspondants et le volume de reconstruction en bas à droite. Résolution des images : 480×640 pixels.

(a) Avec quantification (10 couleurs).

(b) Avec distance entre couleurs (RGB).

(c) Avec manipulation interactive et quantification (10 couleurs).

(d) Problème de l'enveloppe photographique.

(e) Mise en évidence des points à des positions aberrantes.

(f) Problème des points hors du champ de vue des caméras.

FIG. 8.9 – Résultats pour la reconstruction de la cafetière.

effectuée par distance Euclidienne dans l'espace *RGB* et 500 000 points ont été testés par itération. Le résultat est affiché à la figure 8.9 (b).

Méthode interactive : Outre l'ensemble de points initiaux (quelques milliers), nous avons fait exclusivement usage de la région d'intérêt, à l'intérieur de laquelle chaque opération consistait à tester 100 000 points, puis à effectuer une génération par perturbation sur les points acceptés, ensuite à mettre à jour les cartes de profondeurs et enfin à effectuer un nettoyage. Une fusion a été effectuée à un moment où l'ensemble de points était rendu à 775 739 points. Avec une grille de $302 \times 298 \times 301$ voxels, le tout a été réduit à 276 570 points. La comparaison de couleurs est effectuée à l'aide de la quantification. Le résultat est affiché à la figure 8.9 (c).

Les résultats pour les deux méthodes de comparaison de couleurs sont ici très similaires, mais la comparaison par distance entre couleurs est encore une fois plus lente, quoique le tableau 8.4 indique le contraire. Nous rappelons toutefois le grand impact du choix du nombre de points testés par itération (100 000 *vs* 500 000 ici). La méthode interactive demeure la plus efficace.

La figure 8.9 (d) met en évidence une protubérance sur le côté gauche de la cafetière. Ceci est dû à la propriété de l'enveloppe photographique. Il est à noter que seulement neuf images ont été utilisées.

Lorsque nous visualisons la scène depuis un point de vue différent de ceux des images originales, nous apercevons des regroupements de points situés à des positions aberrantes (figure 8.9 (e)). Ceci est dû au fait que la projection du volume de reconstruction n'est pas entièrement incluse dans les images. Or, si un point est généré dans une région du volume de sorte qu'il soit projeté hors des bornes d'une image, il n'est pas considéré visible pour cette dernière. La figure 8.9 (f) illustre le cas en deux dimensions. Le point P n'est pas visible pour les caméras 5 et 6 par information de visibilité sur la scène. Il n'est pas plus visible pour les caméras 1 et 4 (hors du champ de vue). Par contre, il projette dans les caméras 2 et 3 avec la même couleur, ce qui lui donne une plausibilité de 100%, et il est donc accepté. Au lieu de considérer un point qui projette hors des bornes d'une image comme étant non visible, si nous le considérions comme étant masqué (comme pour les masques de rejet), ce problème ne se poserait pas. Cependant, il devient alors impossible de reconstruire des scènes qui ne peuvent pas être entièrement incluses

dans toutes les images. La seule chose à faire est de retirer manuellement ces points. Mais il reste encore que ces derniers, lors de la reconstruction, faussent les cartes de profondeurs. Il est à noter que ce problème est tout aussi présent pour la méthode de sculpture d'espace [KS00].

8.4 Sandwich

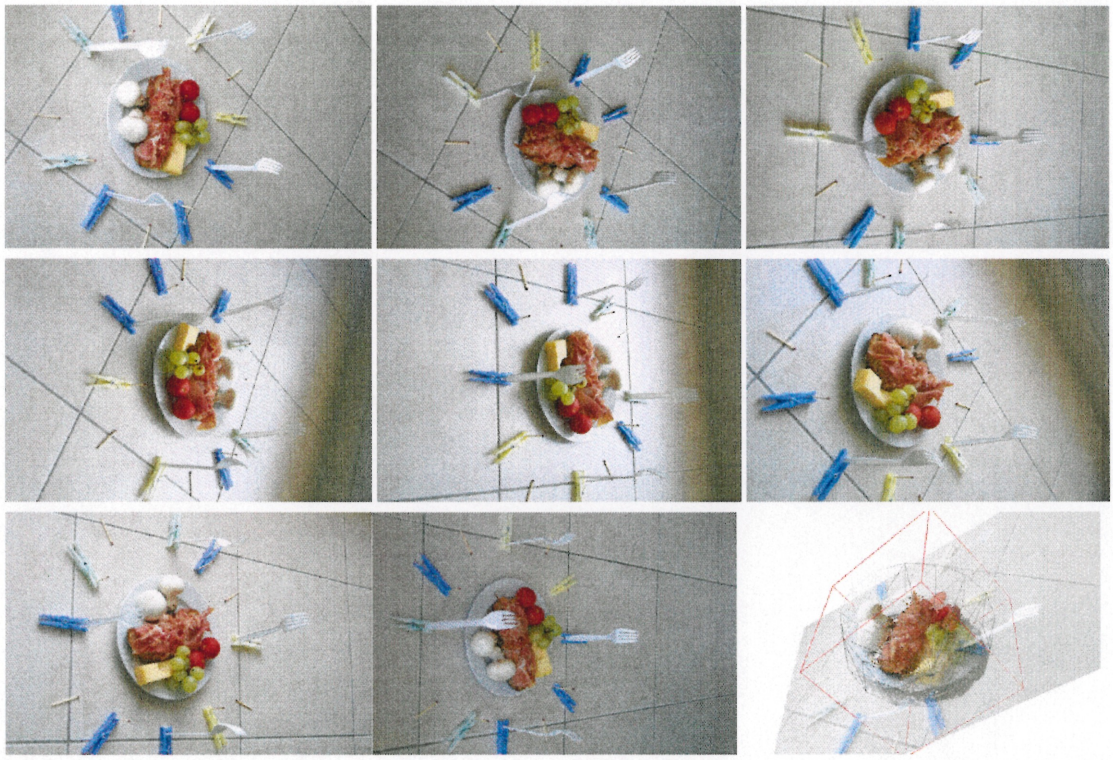
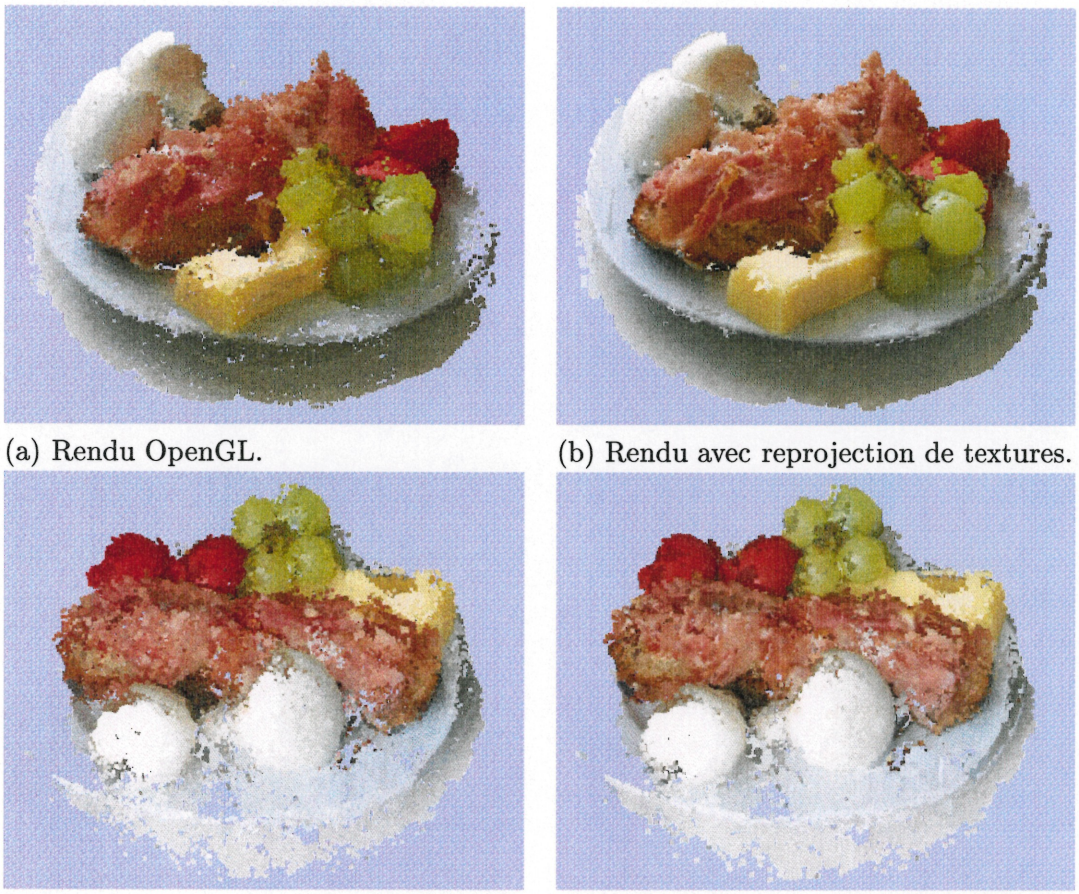


FIG. 8.10 – Les huit images de la scène du sandwich et le volume de reconstruction en bas à droite. Résolution des images : 1 440 × 960 pixels.

La scène du sandwich, dont les images sont présentées à la figure 8.10, présente trois difficultés : la surface spéculaire des fruits et de l'assiette, une géométrie complexe (l'ensemble des raisins et la viande du sandwich), et un nombre limité d'images (huit seulement). Malgré tout, nous obtenons des résultats intéressants, tel que présentés à la figure 8.11.

La reconstruction de la scène a été faite avec une approche interactive similaire à la séance typique de modélisation (section 8.1.1). La comparaison des couleurs a été effectuée avec la quantification (64 couleurs). La séance a requis environ une demi-heure de travail pour produire un ensemble de 284 814 points. Une portion de l'assiette est manquante dans la scène reconstruite. Ceci est dû au mauvais positionnement du



(a) Rendu OpenGL.

(b) Rendu avec reprojection de textures.

(c) Autre point de vue, rendu OpenGL.

(d) Rendu avec reprojection de textures.

FIG. 8.11 – Résultats pour la reconstruction de la scène du sandwich.

volume de reconstruction. La portion manquante est à l'extérieur de ce dernier.

Dans les figures 8.11 (b) et (d), nous pouvons voir le degré de détail atteint à l'aide de la reprojection de textures (section 7.2) comparativement au simple rendu OpenGL (section 7.1) des points. La reprojection de textures réduit l'impact du petit nombre d'images sur la reconstruction.

8.5 Statuette

La scène de la statuette (figures 8.12 et 8.13), tout comme celle de la cafetière (section 8.3), est *a priori* peu complexe. Cependant la texture de sa surface présente plusieurs détails fins et elle est relativement spéculaire. L'occultation a aussi une certaine complexité. Pour la reconstruction, la comparaison de couleurs a été effectuée à l'aide de la quantification (16 couleurs).

Cette scène a posé beaucoup de difficultés pour la méthode de génération de base. Les 50 277 premiers points de la figure 8.14 (a) ont été générés en environ deux heures. Si nous laissons exécuter le processus plus longtemps, ce n'est qu'après huit heures et demie que nous obtenons les 500 001 points de la figure 8.14 (d). Par contre, avec une méthode plus interactive, nous obtenons un résultat satisfaisant de 277 146 points (incluant une fusion) après environ 75 minutes de travail (figure 8.14 (e)). L'ordre dans lequel les régions d'intérêt ont été placées était important dû à la complexité de la visibilité de la statuette. Il s'agit d'une scène où le balayage de plan (section 6.2.4) n'est pas applicable car la scène n'est pas complètement en dehors de l'enveloppe convexe formée par les points de vue des caméras.

La figure 8.14 (f) illustre l'importance des masques de rejet employés pour cette scène, présentés en couleurs inversées aux figures 8.12 et 8.13. Nous n'avons masqué que les trous de la statuette, d'où les petites taches qui se trouvent sur ces masques. Sans eux, la couleur (après quantification) de la surface sur laquelle repose la statuette fait en sorte que des points aberrants apparaissent dans les trous de la statuette. Avec les masques de rejet, certains de ces points sont encore présents. Ceci est dû à l'imprécision des masques (ils ont été faits à la main avec *Gimp*), et possiblement qu'avec un seuil plus élevé que 80% pour le test de plausibilité, d'autres seraient éliminés. Nous aurions pu masquer la région autour de la statuette, mais le volume de reconstruction n'était pas trop grand, comme c'est le cas pour la cafetière (section 8.3), et nous n'avons pas obsersé de points générés à des positions aberrantes autour de la statuette.

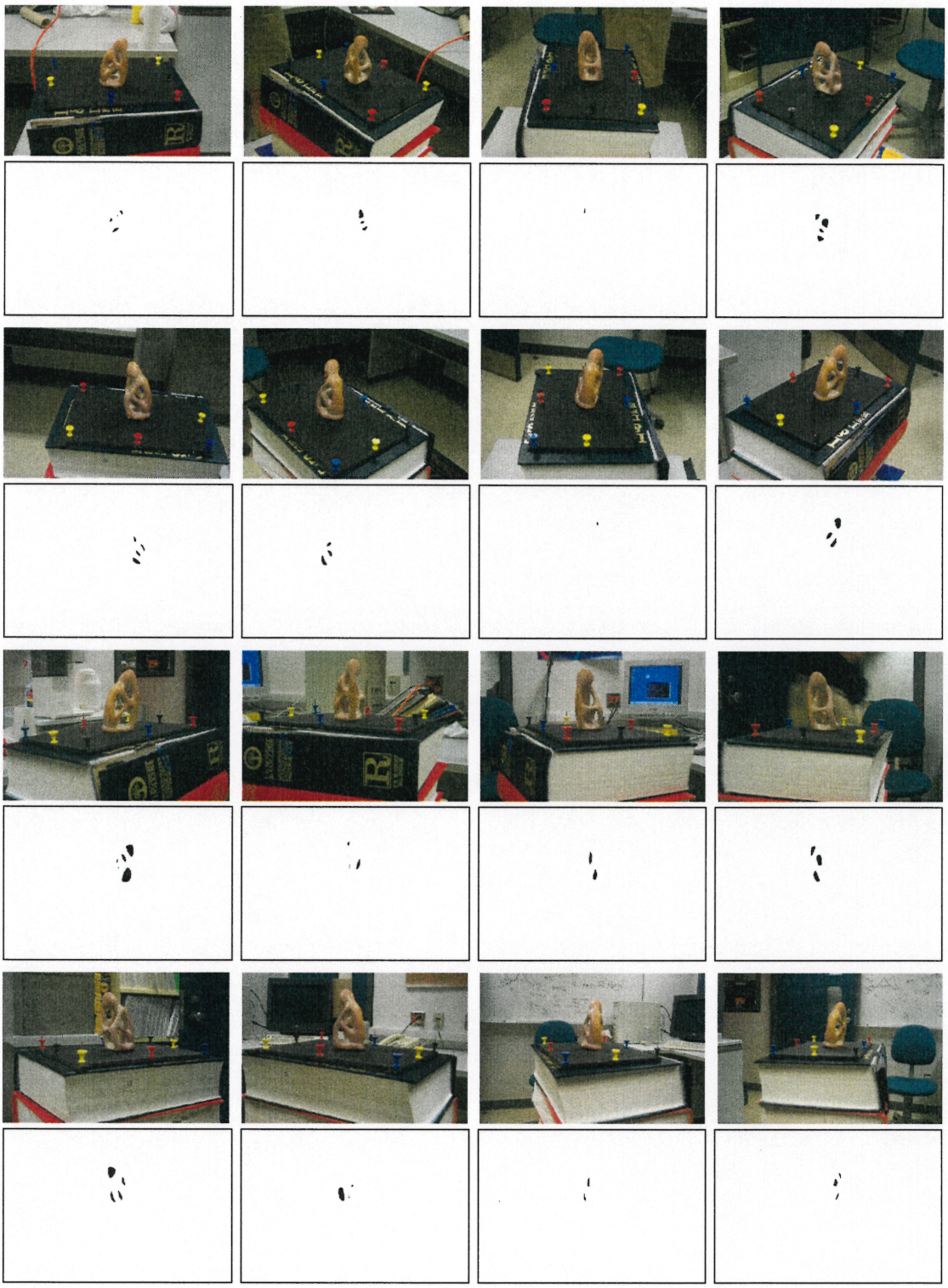


FIG. 8.12 – 16 des 23 images de la scène de la statuette avec leur masque de rejet correspondant (affiché en couleurs inversées). Résolution des images : 1 792 × 1 200 pixels.

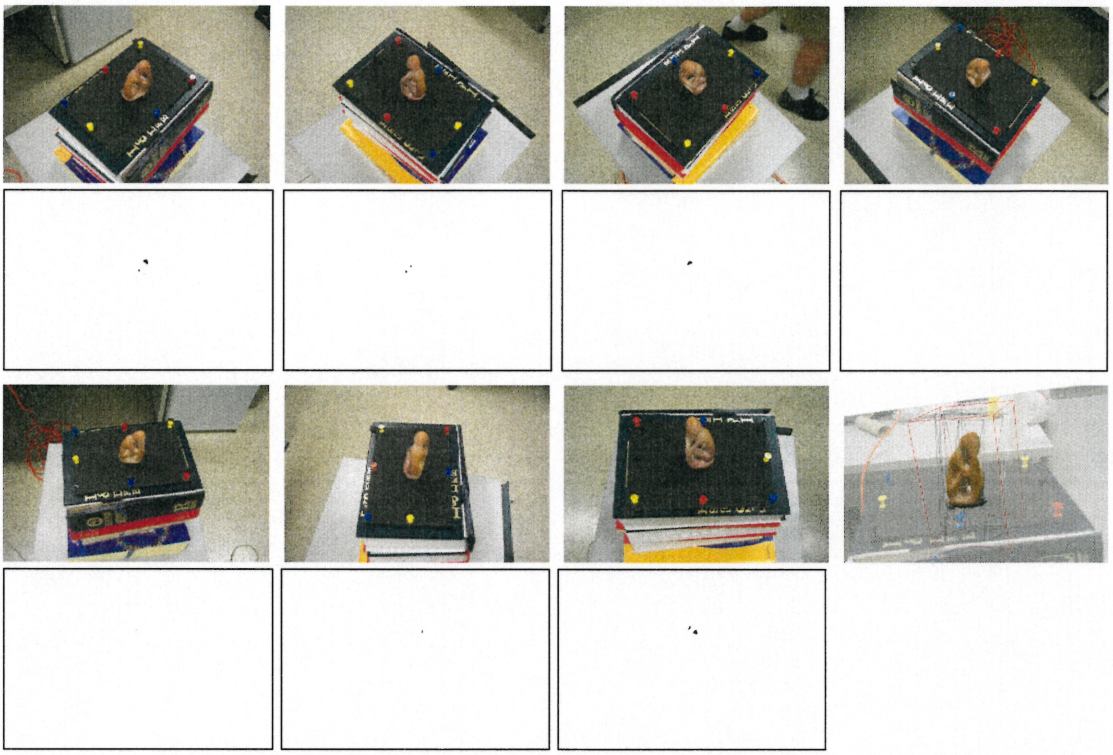
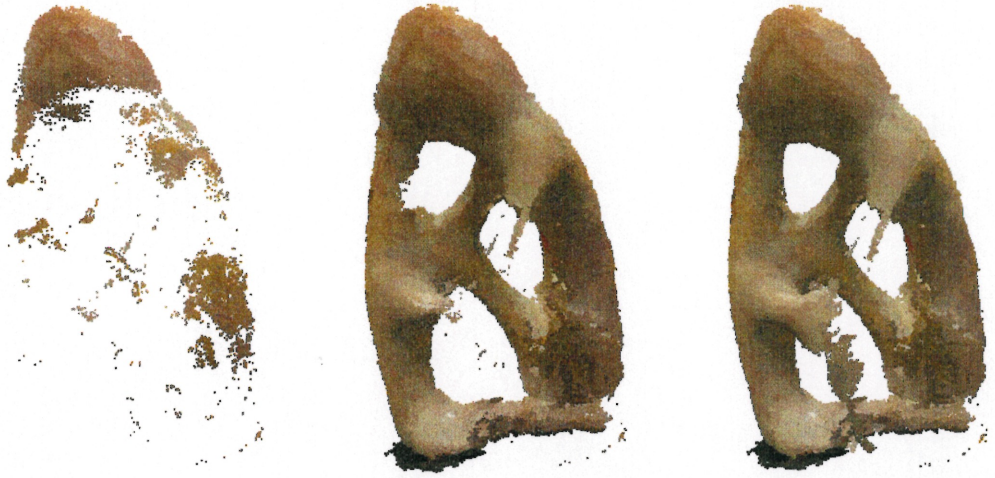


FIG. 8.13 – Les sept dernières images de la scène de la statuette avec leur masque de rejet correspondant (affiché en couleurs inversées) et le volume de reconstruction en bas à droite.

Finalement, étant donné la grande continuité de la surface de la statuette, nous avons effectué un remplissage tridimensionnel (section 6.2.3) sans test de couleurs à partir de l'ensemble de points de la figure 8.14 (a). La densité du remplissage était grande et jusqu'à un maximum de 100 essais étaient permis pour générer un point dans un voxel. L'algorithme a pris un peu moins de deux heures pour produire le résultat de la figure 8.14 (b). L'algorithme n'a pas rempli la statuette au complet à cause de la complexité de la visibilité de cette région. La figure 8.14 (c) montre le résultat après l'exécution d'un autre remplissage (suite au premier) dont le germe est un point dans la région encore vide. Seule la méthode interactive avec la région d'intérêt a permis de bien reconstruire cette région car l'usager peut mieux contrôler l'ordre (spatialement) de génération dans les régions de visibilité plus complexe.



(a) Ensemble initial de 50 277 points.

(b) Après un remplissage tridimensionnel (688 020 points).

(c) Après un remplissage additionnel tridimensionnel (731 656 points).

(d) Avec méthode de base (500 001 points).

(e) Avec méthode interactive (277 146 points).

(f) Comme (e), sans les masques (539 090 points).

FIG. 8.14 – Résultats pour la reconstruction de la statuette.

Chapitre 9

Conclusion

Le besoin d'objets synthétiques de plus en plus réalistes augmente aussi vite que la technologie pour les afficher. Cependant, la création d'objets réalistes complexes est, sauf pour quelques exceptions, une tâche très accaparante. Les photographies d'objets réels sont certes une source très inspirante de réalisme. Avec plusieurs années de recherche fondamentale, les techniques de vision par ordinateur ont produit des résultats impressionnants d'une façon entièrement (ou presque) automatique. Malheureusement, une étude attentive de ces photographies met également en évidence les grands défis auxquels une méthode automatique doit faire face. Les occultations, les textures trop bruitées ou inversement le manque de texture, l'illumination (réflectance, effets spéculaires, *etc.*), et les transparences ne sont que quelques exemples des défis qui nous attendent encore. Plutôt que de développer des solutions à tous ces problèmes, nous proposons d'utiliser les connaissances de l'usager pour réduire les ambiguïtés qui peuvent ainsi survenir dans les images.

Dans ce mémoire, nous avons présenté un système interactif pour extraire des objets complexes d'images. Le processus de reconstruction est similaire aux techniques d'extraction de forme par photo-consistance [SD99, KS00, Dye01, SCMS01], mais avec une approche Monte Carlo, qui converge en général vers la reconstruction produite par la technique de sculpture d'espace [KS00], *i.e.* l'enveloppe photographique. Cependant, cette convergence n'est toujours pas assurée, comme nous l'avons vu pour la scène du buisson synthétique (section 8.2). Elle pourrait l'être avec des techniques de recuit simulé, mais ce ne serait pas une approche raisonnable dans notre contexte. Le problème de convergence se présente généralement dans des scènes avec occultations complexes. Puisqu'au départ nous n'avons pas d'information sur la visibilité, pour être en mesure de générer les premiers points, il faut qu'il y en ait qui soient photo-consistants pour

toutes les images, ce qui n'est pas toujours garanti si les images de points de vue opposés ou très différents n'ont pas de couleurs communes. Nous pouvons alors réduire le seuil d'acceptation pour le test de plausibilité, mais nous risquons d'accumuler plus de points faux. Nous pouvons aussi procéder par sous-ensembles d'images, puis combiner les différents résultats et effectuer un nettoyage.

Traditionnellement, les méthodes d'extraction de forme par photo-consistance emploient une représentation volumique avec des grilles régulières de voxels. Se basant sur plusieurs recherches récentes sur le rendu et la représentation d'objets par points [GD98, PZvBG00, RL00, ZPvBG01, SD01, Lin01, ABCO$^+$01, ZPKG02], nous faisons plutôt usage de celle-ci, ce qui confère plusieurs avantages à notre approche : simplicité, efficacité pour l'affichage, facilité d'intégration dans la plupart des systèmes de rendu, généralement plus économe en mémoire, et aucune contrainte sur la position spatiale ni la taille des échantillons. Ces avantages font en sorte que cette représentation est très appropriée pour les objets de nature complexe.

Avec une représentation par points et nos échantillonnages aléatoires de l'espace, il nous est possible de bien cerner les silhouettes des objets. Cependant, malgré la précision que nous offre cette représentation, tout comme les méthodes d'extraction de forme par photo-consistance, le niveau de détail qu'il est possible d'atteindre pour la reconstruction est limité par la résolution des images. Par contre, employer des images de grande résolution requiert plus de mémoire et plus de temps d'exécution, car l'échantillonnage spatial doit être plus précis pour qu'un point soit photo-consistant. Peu importe la résolution, nous sommes toujours confrontés au problème que la couleur d'un pixel renferme généralement plus d'information qu'elle-même. Il s'agit en fait d'une intégration de l'ensemble des couleurs visibles projetées dans l'espace du pixel. La simple comparaison de couleurs n'est donc pas toujours suffisante pour générer des points qui projettent dans de tels pixels. La quantification tend à réduire les effets de ce problème, mais nous perdons en précision de l'objet reconstruit. Par contre, si nous recherchons seulement un résultat visuellement plaisant, les quelques pertes causées par ce problème peuvent être acceptables.

Une autre force de notre système est le grand contrôle sur le processus de reconstruction offert à l'usager. À tout moment, l'usager peut interactivement changer les valeurs des paramètres des algorithmes implantés. Il peut également guider les méthodes

plus automatiques là où plus de détails sont requis, négliger des régions jugées non nécessaires, ou alors enlever des points faux. Une séance de modélisation typique permet à l'usager d'arrêter le processus, changer une valeur, choisir un algorithme, modifier l'ensemble actuel de points, *etc.*, tout en visualisant interactivement la scène reconstruite en trois dimensions. Ce concept a déjà été exploité avec succès dans plusieurs des meilleurs systèmes de reconstruction architecturale (basée sur des polygones) [DTM96, SWI97, POF98, GMMB00, SRDT01, Ded01, Can, Rea, Phob], bien qu'aucun d'entre eux n'ait abordé l'extraction d'objets complexes comme les exemples que nous avons présentés dans ce mémoire.

L'intégration des concepts d'interactivité et de représentation par points dans notre système offre une plateforme très flexible d'extraction d'objets complexes. Cette flexibilité permet d'expérimenter de façon interactive afin de comprendre les limites de certains algorithmes et de concevoir de nouvelles solutions pour répondre aux défis de la modélisation à partir d'images.

9.1 Améliorations et extensions

Nous avons conçu notre système avec un souci de simplicité et, en général, les solutions les plus simples ont produit de bons résultats en temps raisonnable. Cependant, nous pourrions certainement être plus efficace. Par exemple, l'ensemble des opérations effectuées avec la région d'intérêt profiterait d'une structure spatiale plus avancée des points tridimensionnels [RL00], car en ce moment pour sélectionner les points se trouvant dans cette région, nous testons l'ensemble de tous les points. L'échantillonnage aléatoire uniforme de l'espace n'est pas non plus très efficace, car beaucoup de points hors de la scène sont testés (par exemple, la scène du buisson synthétique, section 8.2). Des approches plus ordonnées, tel le balayage de plan (section 6.2.4) ou des techniques adaptatives (comme elles se sont avérées utiles pour le lancer de rayon), produisent des résultats plus rapidement, mais par contre elles nuisent souvent à l'interactivité. En effet, nous percevons l'allure de la reconstruction de la scène seulement lorsque l'algorithme a terminé (ou presque), alors qu'un échantillonnage aléatoire uniforme produit des résultats intermédiaires plus partiels mais globaux, ce qui permet d'avoir plus rapidement une idée de l'allure finale de la scène reconstruite ou de l'impact de certaines valeurs de seuils. Il faudrait donc optimiser la forme du volume de reconstruction et la distribution de l'échantillonnage aléatoire. La génération par pertubations permet

de générer des points en tenant compte de la distribution actuelle des points, mais elle néglige les régions encore vides. L'intervention de l'usager est généralement cruciale dans ces cas.

Néanmoins, les méthodes de reconstruction tridimensionnelle non interactives peuvent fournir à notre système un ensemble initial de points. La méthode choisie devrait produire des points plus rapidement que notre système, ou encore de meilleure qualité, sinon nous pouvons simplement nous servir de notre système. Par exemple, nous pourrions d'abord effectuer une passe de coloration de voxels en exploitant la cohérence de la grille de voxels. Si les points générés sont bien distribués spatialement dans la scène, ils peuvent fournir des points d'ancrage pour nos techniques de raffinement, en particulier pour l'usage de la région d'intérêt. Ainsi, nous pourrions possiblement fournir à notre système une solution grossière d'un algorithme de stéréovision.

Malgré la simplicité désirée de notre système, il y a un certain nombre de paramètres dont la valeur dépend du choix de l'usager. Ces paramètres incluent le nombre de couleurs pour la quantification, le seuil de plausibilité, les nombres minimum et maximum d'images pour les mises en correspondance, le biais dans les cartes de profondeurs, *etc.* Bien que nous ayons observé des effets pour plusieurs de ces paramètres, il faudrait développer des tests plus exhaustifs ou systématiques pour cerner plus précisément leur impact ou leur importance. Dans le contexte de ce travail et faute de temps, nous procédons sur une base plus intuitive que formelle.

Avec la nature quasi-indépendante de chaque point, nous pourrions aussi songer à faire usage du parallélisme dans nos algorithmes. En ce moment, le partage d'une structure commune pour l'emmagasinage des points fait en sorte qu'un seul processus à la fois peut accéder à cette structure. Cependant, rien n'empêche de maintenir une liste de points par processus et de temps à autre effectuer une synchronisation pour mettre à jour des cartes de profondeurs, puisque cette dernière opération n'est faite que sporadiquement. D'autres algorithmes, comme le remplissage de vue ou le remplissage tridimensionnel, profiteraient aussi beaucoup du parallélisme.

Plusieurs opérations requises pour notre système doivent être effectuées de façon externe, telle la création des masques de rejet, ou avec plus d'une séance de modélisation, telle la reconstruction par sous-ensembles d'images (une séance par sous-ensemble). Pour permettre à l'usager d'être plus efficace ou d'observer directement l'effet de modification

des masques ou du choix des points de vue, il faudrait intégrer le tout dans notre système. La sélection des sous-ensembles pourrait également être faite de façon plus automatique. Toutes ces modifications seraient plutôt faciles à intégrer dans notre système.

Dans son état actuel, le réalisme des scènes reconstruites par notre système est certes prometteur, mais il n'est pas encore entièrement satisfaisant pour des applications photoréalistes. À la base, nous produisons des résultats comparables à ceux des algorithmes automatiques. L'intervention de l'usager augmente davantage le niveau de qualité, mais atteindre entièrement le photoréalisme peut exiger énormément de travail, si toutefois cela est possible. En plus d'être limités par la résolution des images originales, nous sommes confrontés à l'extraction d'autres facteurs que la géométrie et les couleurs pour atteindre entièrement le photoréalisme. Voici un ensemble d'aspects qu'il faudrait considérer avec plus d'attention :

- De façon générale, nous pourrions profiter de la plupart des informations exploitées par les méthodes de reconstruction dites *forme extraite de X*, où X peut être la silhouette, le mouvement, l'illumination, le focus/défocus ou la texture. Par exemple, avec le focus/défocus, il serait intéressant d'essayer d'extraire des modèles géométriques de divers niveaux de détail selon l'état du focus.
- Notre méthode de comparaison de couleurs assume des modèles de réflectance limités, près du modèle lambertien, et une couverture complète et uniforme de chaque pixel. Les variations d'illumination devraient être intégrées dans notre test de comparaison de couleurs. Du même coup, nous extrairions partiellement les paramètres de ces modèles d'illumination pour de nouvelles conditions d'éclairage. Les régions dans l'ombre pourraient également nous fournir de l'information sur la couleur des surfaces sans illumination directe. L'extraction de normales et la direction des sources de lumière dominantes serait également nécessaire.
- La transparence, la réflexion et la réfraction posent beaucoup de problèmes de mise en correspondance car les couleurs des surfaces ayant de telles propriétés dépendent du point de vue. Nous pourrions aider le processus de reconstruction en plaçant des primitives synthétiques simples dans ces régions de la scène et faire usage de cartes d'environnement (*environment maps*) [BN76]. Il faudrait cependant faire attention pour l'acquisition des images et des cartes d'environnement, car typiquement, pour acquérir ces dernières, nous plaçons une sphère miroir à la

place de l'objet et nous photographions la sphère selon les mêmes points de vue que nous photographions l'objet. Pour s'assurer que les points de vue et la position de l'objet et de la sphère soient identiques, il faudrait faire usage de multiples caméras sur trépied.

- Une limite inhérente de la représentation par point est que nous sommes limités sur le zoom que nous pouvons faire sur la scène. À un certain moment, les points sont trop espacés et les aggrandir produit des surfaces discrétisées trop grossières. Il faudrait à ce moment soit effectuer une triangulation locale, soit procéder avec des techniques de raffinement d'échantillonnage dynamique silimaires à celle de Stamminger et Drettakis [SD01].

Nous pourrions aussi nous intéresser à la capture de mouvement à partir d'images. En supposant que nous disposons de multiples caméras pour prendre les images à chaque instant de façon simultanée, nous voudrions extraire l'animation d'un objet. Pour ce faire, nous pourrions tout simplement lancer notre méthode de reconstruction à chaque moment (*frame*), ce qui exigerait beaucoup de travail, ou nous pourrions exploiter la cohérence temporelle entre chaque moment en nous basant sur la reconstruction du moment précédent. Si l'objet est rigide, nous pourrions aussi le reconstruire une seule fois au premier moment et ensuite ajuster sa position selon les moments subséquents.

Nous pourrions enfin nous intéresser à la reconstruction d'objets volumiques tels la fumée ou les nuages, ou alors nous baser sur l'imagerie médicale (rayons X). Encore une fois, de multiples caméras seraient nécessaires. Le problème de ces objets est leur transparence qui varie selon leur densité. Le seuil d'acceptation pour les points pourrait alors être défini plutôt en terme d'accumulation de densité lors de la projection des points dans les pixels d'un tampon d'accumulation pour chaque image.

Bibliographie

[ABCO+01] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin et C. T. Silva. « Point set surfaces ». *IEEE Visualization 2001*, pages 21–28, octobre 2001.

[Ado] Adobe. http ://www.adobe.com/.

[Ali] Alias|Wavefront. http ://www.aliaswavefront.com/.

[Aya91] Nicolas Ayache. *Artificial Vision for Mobile Robots — Stereo-vision and Multisensor Perception.* MIT Press, Cambridge, MA, 1991.

[BBM+01] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler et Michael Cohen. « Unstructured Lumigraph Rendering ». In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 425–432, août 2001.

[Bli82] James T. Blinn. « Light Reflection Functions for Simulation of Clouds and Dusty Surfaces ». In *Proceedings of SIGGRAPH 82*, Computer Graphics Proceedings, Annual Conference Series, pages 21–29, juillet 1982.

[Blo97] Jules Bloomenthal, éditeur. *Introduction to Implicit Surfaces.* Morgan-Kaufmann, 1997.

[BN76] J. F. Blinn et M. E. Newell. « Texture and Reflection in Computer Generated Images ». *Communications of ACM*, volume 19, numéro 10, pages 542–547, octobre 1976.

[Can] Canoma. http ://www.canoma.com/.

[CB92] R. Cipolla et A. Blake. « Surface Shape from the Deformation of Apparent Contours ». *International Journal of Computer Vision*, volume 9, numéro 2, pages 83–112, novembre 1992.

[CHP+79] C. Csuri, R. Hackathorn, R. Parent, W. Carlson et M. Howard. « Towards an Interactive High Visual Complexity Animation System ». In *Proceedings*

of SIGGRAPH 79, Computer Graphics Proceedings, Annual Conference Series, pages 289–299, août 1979.

[Cor02] Immersion Corporation. « MicroScribe ». http ://www.immersion.com/products/3d/capture/msinfo.shtml, 2002.

[Cyb] Cyberware. http ://www.cyberware.com/.

[Ded01] Sébastien Dedieu. *Adaptation d'un système de reconstruction de modèles numériques 3D à partir de photographies face aux connaissances de l'utilisateur.* Thèse de doctorat, Université de Bordeaux I, décembre 2001.

[DFL02] Paul Debevec, Nickson Fong et Dan Lemmon. « Image-Based Lighting ». SIGGRAPH 2002 Course #5, juillet 2002.

[DHL+98] Oliver Deussen, Pat Hanrahan, Bernd Lintermann, Radomír Měch, Matt Pharr et Przemyslaw Prusinkiewicz. « Realistic Modeling and Rendering of Plant Ecosystems ». In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 275–286, août 1998.

[DM97] Paul E. Debevec et Jitendra Malik. « Recovering High Dynamic Range Radiance Maps from Photographs ». In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 369–378, août 1997.

[DTM96] Paul E. Debevec, Camillo J. Taylor et Jitendra Malik. « Modeling and Rendering Architecture from Photographs : A Hybrid Geometry- and Image-Based Approach ». In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 11–20, août 1996.

[Dye01] Charles. R. Dyer. « Volumetric Scene Reconstruction from Multiple Views ». In *Foundation of Image Understanding*, pages 469–489, 2001.

[EL93] J. Ens et P. Lawrence. « An Investigation of Methods for Determining Depth from Focus ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 15, numéro 2, pages 97–108, 1993.

[Far] Faro. http ://www.faro.com.

[Fau93] Olivier Faugeras. *Three-Dimensional Computer Vision : A Geometric Viewpoint.* MIT Press, Cambridge (MA), 1993.

[FR86] Alain Fournier et William T. Reeves. « A Simple Model of Ocean Waves ». In *Computer Graphics (Proceedings of SIGGRAPH 86)*, volume 20, pages 75–84, août 1986.

[FvDFH96] James D. Foley, Andries van Dam, Steven K. Feiner et John F. Hughes. *Computer Graphics, Principles and Practice.* Addison-Wesley, second edition in c édition, 1996.

[GD98] J. P. Grossman et William J. Dally. « Point Sample Rendering ». In *9th Eurographics Workshop on Rendering*, pages 181–192, 1998.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski et Michael F. Cohen. « The Lumigraph ». In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 43–54, août 1996.

[Gim] The Gimp. http ://www.gimp.org/.

[GMMB00] E. Guillou, D. Meneveaux, E. Maisel et K. Bouatouch. « Using Vanishing Points for Camera Calibration and Coarse 3D Reconstruction from a Single Image ». *The Visual Computer*, volume 16, numéro 7, pages 396–410, 2000.

[GSS99] Igor Guskov, Wim Sweldens et Peter Schröder. « Multiresolution Signal Processing for Meshes ». In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 325–334, août 1999.

[GW00] Abbas El Gamal et Brian Wandell. « Programmable Digital Camera Project ». http ://www-isl.stanford.edu/~abbas/group/imaging.shtml, août 2000.

[Ham01] Hamamatsu. « High Dynamic Range Streak Camera C7700 ». http ://usa.hamamatsu.com/sys-streak/c7700/default.htm, 2001.

[HB89] B. Horn et M. Brooks. *Shape from Shading.* MIT Press, Cambridge, MA, 1989.

[HH90] Pat Hanrahan et Paul Haeberli. « Direct WYSIWYG Painting and Texturing on 3D Shapes ». In *Proceedings of SIGGRAPH 90*, Computer Graphics Proceedings, Annual Conference Series, pages 215–223, 1990.

[Hit00] Hitashi. « VK-S454 Wide Dynamic Range Colour Zoom Camera ». http ://www.hitachi.ca/dig_med/mm/dig_camera/halp-ZZZXZ9VP1NC.html, 2000.

[KC89] K. I. Kanatani et T. C. Chou. « Shape from Texture : General Principle ». *Artificial Intelligence*, volume 38, numéro 1, pages 1–48, 1989.

[KCVS98] Leif Kobbelt, Swen Campagna, Jens Vorsatz et Hans-Peter Seidel. « Interactive Multi-Resolution Modeling on Arbitrary Meshes ». In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 105–114, juillet 1998.

[KS00] Kiriakos N. Kutulakos et Steve M. Seitz. « A Theory of Shape by Space Carving ». *International Journal of Computer Vision*, volume 38, numéro 3, pages 199–218, 2000.

[KSK98] Reinhard Klette, Karsten Schluns et Andreas Koschan. *Computer Vision : Three-Dimensional Data from Images.* Springer, 1998.

[LH81] H. C. Longuet-Higgins. « A Computer Algorithm for Reconstructing a Scene from Two Projections ». *Nature*, volume 293, numéro 10, pages 133–135, 1981.

[LH96] Marc Levoy et Pat Hanrahan. « Light Field Rendering ». In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 31–42, août 1996.

[Lin68] Aristid Lindenmayer. « Mathematical Models for Cellular Interaction in Development, Parts I and II ». *Journal of Theoretical Biology*, volume 18, pages 280–315, 1968.

[Lin01] Lars Linsen. « Point Cloud Representation ». Rapport technique 2001-3, Fakultät für Informatik, Universität Karlsruhe, 2001.

[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 *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 131–144, 2000.

[LW85] Marc Levoy et Turner Whitted. « The Use of Points as a Display Primitive ». Rapport technique 85-022, University of North Carolina Computer Science Department, Chapell Hill, NC 27514, janvier 1985.

[Min02] Minolta. « 3D Digitizers ». http ://www.minolta3d.com/, 2002.

[MKM89] F. Kenton Musgrave, Craig E. Kolb et Robert S. Mace. « The Synthesis and Rendering of Eroded Fractal Terrains ». In *Computer Graphics (Proceedings of SIGGRAPH 89)*, volume 23, pages 41–50, juillet 1989.

[MPN+02] Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul Beardsley, Remo Ziegler et Leonard McMillan. « Imaged-Based 3D Photography using Opacity Hulls ». In *Proceedings of SIGGRAPH 2002*, Computer Graphics Proceedings, Annual Conference Series, pages 427–434, juillet 2002.

[*n*VIDIA] *n*VIDIA. http ://www.nvidia.com/.

[OHHM02] Marc Olano, John C. Hart, Wolfgang Heidrich et Michael McCool. *Real-Time Shading*. A. K. Peters, juillet 2002.

[PG01] Mark Pauly et Markus Gross. « Spectral Processing of Point-Sampled Geometry ». In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 379–386, août 2001.

[Phoa] Kodak PhotoCD. http ://www.kodak.com/daiHome/products/photoCD.shtml.

[Phob] PhotoModeler. http ://www.photomodeler.com/.

[PL90] Przemyslaw Prusinkiewicz et Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer–Verlag, New-York, 1990.

[PM01] Yaov I. H. Parish et Pascal Müller. « Procedural Modeling of Cities ». In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 301–308, août 2001.

[POF98] Pierre Poulin, Mathieu Ouimet et Marie-Claude Frasson. « Interactively Modeling with Photogrammetry ». In *Proceedings of Eurographics Workshop on Rendering 98*, pages 93–104, juin 1998.

[Pol] Polhemus. « FastSCAN 3D Handheld Laser Scanner ». http ://www.polhemus.com/fastscan.htm.

[Pri02] Keith Price. « Annotated Computer Vision Bibliography : Table of Contents ». http ://iris.usc.edu/Vision-Notes/bibliography/contents.html, août 2002.

[PZvBG00] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar et Markus Gross. « Surfels : Surface Elements as Rendering Primitives ». In *Proceedings of*

SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series, pages 335–342, juillet 2000.

[RB93] Jarek R. Rossignac et Paul Borrel. « Multi-Resolution 3D Approximation for Rendering Complex Scenes ». In *Geometric Modeling in Computer Graphics*, pages 455–465, juin 1993.

[Rea] RealViz. http ://www.realviz.com/.

[Ree83] Wiliam T. Reeves. « Particle Systems — A Technique for Modeling a Class of Fuzzy Objects ». *Computer Graphics*, volume 17, numéro 3, pages 359–376, juillet 1983.

[RHHL02] Szymon Rusinkiewicz, Olaf Hall-Holt et Marc Levoy. « Real-Time 3D Model Acquisition ». In *Proceedings of SIGGRAPH 2002*, Computer Graphics Proceedings, Annual Conference Series, pages 438–446, juillet 2002.

[RL00] Szymon Rusinkiewicz et Marc Levoy. « QSplat : A Multiresolution Point Rendering System for Large Meshes ». In *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 343–352, juillet 2000.

[Sam90] Hanan Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, MA, 1990.

[SCMS01] G. Slabaugh, W. B. Culbertson, T. Malzbender et R. Schafer. « A Survey of Volumetric Scene Reconstruction Methods from Photographs ». In *Volume Graphics 2001, Joint Proceedings of IEEE Transaction on Computer Visualization and Graphics and Eurographics Workshop*, pages 151–162, juin 2001.

[SD99] Steve M. Seitz et Charles. R. Dyer. « Photorealistic Scene Reconstruction by Voxel Coloring ». *International Journal of Computer Vision*, volume 35, numéro 2, pages 151–173, 1999.

[SD01] Marc Stamminger et George Drettakis. « Interactive Sampling and Rendering for Complex and Procedural Geometry ». In *Rendering Techniques 2001 (Proceedings of the Eurographics Workshop on Rendering 01)*, 12th Eurographics Workshop on Rendering, pages 151–162, 2001.

[Smi84] Alvy Ray Smith. « Plants, Fractals and Formal Languages ». In *Proceedings of SIGGRAPH 84*, Computer Graphics Proceedings, Annual Conference Series, pages 1–10, juillet 1984.

[Sof] Softimage. http ://www.softimage.com/.

[SP98] S. Sullivan et J. Ponce. « Automatic Model Construction and Pose Estimation from Photographs Using Triangular Splines ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 20, numéro 10, pages 1091–1096, octobre 1998.

[Sph02] SpheronVR. « High Dynamic Range Imaging with SpheronVR and PanoCam ». http ://www.spheronvr.com/products/HDRI/hdri.html, juillet 2002.

[SRDT01] Ilya Shlyakhter, Max Rozenoer, Julie Dorsey et Seth Teller. « Reconstructing 3D Tree Models from Instrumented Photographs ». *IEEE Computer Graphics and Applications*, volume 21, numéro 3, pages 53–61, 2001.

[SVZ00] Dan Snow, Paul Viola et Ramin Zabih. « Exact Voxel Occupancy with Graph Cuts ». In *Computer Vision and Pattern Recognition Conference*, volume 1, pages 345–352, 2000.

[SWI97] Yoichi Sato, Mark D. Wheeler et Katsushi Ikeuchi. « Object Shape and Reflectance Modeling from Observation ». In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 379–388, août 1997.

[Sze93] R. Szeliski. « Rapid Octree Construction from Image Sequences ». *Computer Vision, Graphics and Image Processing*, volume 58, numéro 1, pages 23–32, juillet 1993.

[Tau95] Gabriel Taubin. « A Signal Processing Approach To Fair Surface Desing ». In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 351–358, août 1995.

[TF87] G. Toscani et O. Faugeras. « Structure from Motion Using the Reconstruction and Reprojection Technique ». In *IEEE Computer Society Workshop on Computer Vision*, pages 345–348, novembre 1987.

[TK92] C. Tomasi et T. Kanade. « Shape and Motion from Image Streams Under Ortography : a Factorisation Method ». *International Journal of Computer Vision*, volume 9, numéro 2, pages 137–154, novembre 1992.

[TV98] Emanuele Trucco et Alessandro Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall, Upper Saddle River, New Jersey 07458, 1998.

[WNDS97] Mason Woo, Jackie Neider, Tom Davis et Dave Shreiner. *OpenGL Programming Guide : the Official Guide to Learning OpenGL, Version 1.2.* Addison-Wesley, 1997.

[WS82] G. Wyszecki et W. Stiles. *Color Science : Concepts and Methods, Qunatitative Data and Formulae.* Wiley, New York, deuxième édition, 1982.

[Zhe94] J. Y. Zheng. « Acquiring 3-D Models from a Sequence of Contours ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 16, numéro 2, pages 163–178, février 1994.

[ZPKG02] Matthias Zwicker, Mark Pauly, Oliver Knoll et Markus Gross. « Pointshop 3D : An Interactive System for Point-Based Surface Editing ». In *Proceedings of SIGGRAPH 2002*, Computer Graphics Proceedings, Annual Conference Series, pages 323–329, juillet 2002.

[ZPvBG01] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar et Markus Gross. « Surface Splatting ». In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 371–378, août 2001.

[ZTCS99] R. Zhang, P. S. Tsai, J. Cryer et M. Shah. « Shape from Shading : A Survey ». *IEEE Transaction on Pattern Analysis and Machine Intelligence*, volume 21, numéro 8, pages 690–706, août 1999.