Université de Montréal

Excursions en rendu par champ de lumière: champ de visibilité et ré-illumination

par

Martin Blais

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

Octobre 1998

Université de Montréal

Faculté des études supérieures

Ce mémoire de maîtrise intitulé

Excursions en rendu par champ de lumière: champ de visibilité et ré-illumination

présenté par

Martin Blais

a été évalué par un jury composé des personnes suivantes :

Président :	Jean Meunier
Directeur de recherche :	Pierre Poulin
Membre :	François Major

Sommaire

Récemment, on observe dans le domaine du rendu une explosion de la prolifération des techniques de rendu basé sur des images (*image-based rendering*). Notamment, quelques techniques d'échantillonnage dans le domaine des lignes dans l'espace tridimensionnel ont été proposées pour capturer le champ de lumière émanant d'un objet et pour en reconstruire certaines vues (en le ré-échantillonnant). Ce rendu par champ de lumière a l'avantage de pouvoir capturer des effets arbitrairement complexes et de nous permettre de les recréer efficacement. Par contre, plusieurs problèmes restent encore à résoudre pour permettre l'utilisation pratique de ces structures dans les systèmes de rendu. Nos travaux de recherche explorent deux volets du rendu par champ de lumière, pour tenter d'en étendre les possibilités.

Dans un premier volet, nous proposons un algorithme de visibilité utilisant des structures d'échantillonnage en quatre dimensions similaires à celles utilisées en rendu par champ de lumière (paramétrisation à deux plans), pour faire un précalcul de visibilité qui nous permettra d'éliminer de manière approximative les surfaces cachées en temps réel.

Dans un deuxième volet, nous explorons la ré-illumination de champ de lumière, dans le but d'inscrire des objets ainsi représentés à l'intérieur de scènes synthétiques ou réelles. Pour ce faire, nous étendons la structure d'échantillonnage du champ de lumière pour contenir de l'information limitée sur la géométrie, ce qui nous permettra de faire les calculs d'illumination pertinents.

Mots-clés :

Infographie 3D, image de synthèse, ré-illumination, rendu basé sur des images, champ de lumière, visibilité, champ de visibilité.

Table des matières

Liste des tableaux

Table des figures

Remerciements

Des gens

Je tiens d'abord à remercier mes concepteurs pour leur support durant cette entreprise. Sans eux, rien n'aurait été possible (forcément).

Je dois témoigner gratitude aux autres membres du laboratoire pour avoir supporté mes extravagances et mon attitude, ils savent bien tous que j'ai toujours raison. Plus particulièrement, je remercie Cyriaque Kouadio et Marie-Claude Frasson pour leur soutien moral, ainsi que pour m'avoir initié (i.e. forcé) à l'entraînement physique. Je dois aussi remercier Patrick Fournier et Mathieu Ouimet pour la dérision quotidienne ainsi que pour la production du film court "reggae night". Normand Brière et Éric Plante m'ont entretenu de nombreuses discussions enrichissantes reliées à la recherche en infographie et j'en suis reconnaissant.

À UBC, mes gratitudes vont à Alain Fournier et Kellogg Booth pour m'avoir si bien accueilli dans leur laboratoire, où j'ai passé quatre mois très intéressants. Je remercie Alain et Paul Lalonde pour les discussions qui m'auront aidé à faire avancer mon projet sur la ré-illumination. Je suis reconnaissant à Lifeng Wang et Robert Scharein pour avoir fourni des modèles d'objets pour mon travail. Tous les membres du laboratoire Imager auront contribué à mon plaisir quotidien par leur grand intérêt à la discussion argumentative. Je tiens particulièrement à remercier Marcelo Walter et Gene Lee pour avoir supporté mes attaques répétitives avec les *NURF weapons* ainsi que pour leur amitié et leur support moral.

Finalement, je tiens à exprimer mes profonds remerciements à Pierre Poulin, mon directeur de recherche, pour m'avoir introduit aux concepts de "l'énergie rouge", du verbe "caractégoriser", pour m'avoir fait croire quelques instants en l'expression d'une ligne dans l'es-

pace sous la forme $Ax + By + Cz + D = 0$, ainsi que pour la multitude de discussions à propos de divers sujets infographiques qui m'auront apporté de connaissances précieuses, et donné un esprit plus critique face à la recherche. Je dois aussi le louer pour m'avoir laissé la liberté de bidouiller[1] de multiples autres projets et infographies ésotériques en marge de mon travail officiel, ainsi que pour son support face à mon échange à UBC. À ce jour, je n'arrive toujours pas à comprendre comment il a pu endurer mon caractère intempestif pendant bientôt deux ans. Mais je tiens tout particulièrement à le remercier pour sa grande disponibilité, sa confiance et son amitié, qui m'ont souvent été nécessaires lors de ces deux années.

Des organismes

Ces travaux de recherche ont été subventionnés par le Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (Fonds FCAR), et j'en suis très reconnaissant. J'aimerais aussi mentionner mon appréciation pour Taarna Studios, pour avoir gardé contact de par leurs invitations et activités durant mes études.

Je dois aussi remercier le laboratoire d'infographie de Stanford pour avoir rendu disponible leur logiciel de rendu par champ de lumière sur le *web*, ainsi que leurs données. Quelques fonctions de leur code furent utilisées à l'intérieur de mon implémentation.

Des machines

Lorsque j'ai entamé cette maîtrise, c'était un peu une excuse pour prendre le temps de lire des articles et des bouquins d'infographie pendant deux ans. Je suis bien content d'avoir pu réaliser cet objectif et d'avoir réussi tout le reste en plus. Je tiens donc à remercier chaleureusement la photocopieuse du département pour m'avoir supporté dans cette entreprise.

[1] "Bidouiller" est la traduction officielle de l'Office de la Langue Française pour le verbe : *to hack* [OLF98].

Chapitre 1

Introduction

"The bright outlook for memory technology bodes well for mass-memory graphics. [. . .] The challenge of this approach to computer graphics is to cope successfully with the problems posed by the discrete nature of image space scene representations."

—Lance Williams,
"Casting Curved Shadows on Curved Surfaces", 1978

Comme le prédisait Lance Williams il y a vingt ans, l'explosion rapide de la quantité de mémoire disponible à un ordinateur nous permet d'utiliser des algorithmes gourmands en stockage de données. Le champ du rendu basé sur des images est un signe de l'émergence d'une nouvelle classe de techniques qui se présente aujourd'hui comme une solution plus accessible face au problème du rendu et ce, à cause de la croissance de la taille mémoire disponible. Entre autres, le rendu par champ de lumière est une méthode particulièrement gourmande en espace mémoire. Cette méthode consiste à échantillonner le champ plénoptique dans l'espace et à utiliser cet échantillonnage comme base au rendu de nouvelles images. Par contre, bien que cette technique apporte des éléments nouveaux et originaux au niveau du rendu, elle possède de sérieuses limites dans un contexte plus général. Dans ce mémoire nous explorerons deux directions de recherche pour tenter d'élargir l'étendue de cette méthode.

1.1 Contexte

La multitude de problèmes reliés à l'aliassage en infographie sont assez bien connus aujourd'hui (voir par exemple les travaux de Glassner [Gla95], Mitchell et Netravali [MN88] ou Heckbert [Hec86a] [Hec86b]), et l'utilisation de structures bidimensionnelles comme support au rendu est maintenant courante. On peut penser, par exemple, aux cartes d'ombres [Wil78], aux cartes d'illumination [Hec90], aux textures de relief [Bli78], ainsi qu'au tampon de profondeur hiérarchique [GK93].

Toutes ces techniques (et bien d'autres) utilisent des images (sous forme de grille ou de *quadtree*) pour stocker des résultats intermédiaires nécessaires à leur algorithme respectif. Elles sont donc toutes sujettes aux problèmes d'aliassage en deux dimensions, et plusieurs techniques d'échantillonnage et de ré-échantillonnage ont été développées pour pallier à ces problèmes. Même si les limites inhérentes exprimées par le théorème de Nyquist posent des contraintes sur la qualité des approximations que l'on peut obtenir en espace image, la grande quantité de stockage à notre disposition nous permet d'utiliser des tailles d'images qui souvent réduisent les aléas reliés à l'aliassage.

Avec l'évolution des techniques de rendu, on a commencé à rendre des effets plus subtils qui requièrent des structures avec un nombre plus grand de dimensions. On peut penser au complexe [DDP96] ou au squelette de visibilité [DDP97], qui stocke sous forme d'un graphe tous les changements de visibilité entre les polygones [GM90], ou encore à la capture et l'utilisation de BRDFs[1], qui sous sa forme échantillonnée, consiste en un tableau de quatre dimensions ou plus [NRH$^+$77] [DF97]. Par ailleurs, l'utilisation de structures de stockage de tailles appréciables est devenue de plus en plus acceptable dans ce domaine. Par exemple, les techniques de calcul de radiosité par élément finis [CW93b] ainsi que le *photon map* [Jen95] créent et conservent de grosses structures en mémoire dans le but d'évaluer l'illumination globale dans une scène.

Plus récemment, on a commencé à utiliser des images comme matériel de base pour créer des rendus tridimensionnels, car ces dernières peuvent capturer des effets difficiles à rendre

[1] Une *Bidirectional Reflection Distribution Function* (fonction bidirectionnelle de distribution de la réflexion) caractérise les propriétés de réflexion d'une surface.

en simulant la propagation de la lumière sur des objets difficiles à modéliser. C'est ce qu'on appelle aujourd'hui le champ du rendu basé sur des images (en anglais : *image-based rendering*). Bien qu'en général moins précises à cause de la nature discrète des images utilisées, ces techniques offrent aussi l'avantage qu'il est assez rapide (temps constant) de faire des manipulations sur les pixels d'images pour obtenir des effets tridimensionnels. Conséquemment, un attrait du rendu basé sur des images repose dans la possibilité de faire des rendus à des vitesses interactives, en considérant par contre un coût important en terme d'utilisation de la mémoire.

Comme une image peut être considérée comme un tableau bidimensionnel d'échantillons de radiance incidents à un plan, l'idée d'échantillonner la radiance de façon plus systématique est apparue en même temps comme une extension du rendu basé sur des images [LH96] [GGSC96] et comme une technique pour simuler la propagation dans l'espace de la lumière échantillonnée [LF96]. Ces travaux de recherche doivent traiter de l'échantillonnage et de la représentation de toute la radiance qui se propage dans une scène, c'est-à-dire du *champ de lumière*, une fonction à cinq dimensions qui dépend de la position et de la direction dans l'espace. Évidemment, les structures de données ainsi créées occupent des tailles mémoire titanesques.

Par contre, elles offrent l'avantage qu'on puisse capturer des effets subtils à partir du monde réel, et les reproduire directement. Ceci est surtout intéressant pour des effets impossibles à simuler avec les méthodes connues. Le compromis qu'on doit malheureusement accepter se situe dans la nature discrète de la représentation qu'on utilise pour traiter les champs de lumière. C'est ainsi que l'histoire se répète, car aujourd'hui aussi, on peut espérer dans le futur avoir à notre disposition des tailles mémoire qui rendraient acceptable de stocker ce genre de structures de données, s'il en vaut la peine.

Mais là réside la vraie question : quels sont les problèmes que l'on peut résoudre avec ces représentations de la lumière, et quelles sont les limites de ces techniques ? Nous tenterons d'y répondre partiellement dans le présent ouvrage. Nous explorerons quelques possibilités d'extensions au rendu par champ de lumière tel que présenté par Levoy et Hanrahan [LH96]. Voilà pourquoi ce mémoire traite "d'excursions" en rendu par champ de lumière.

1.2 Motivation

Pour que le rendu par champ de lumière devienne utilisable hors du contexte limité de la recherche, il est primordial d'élargir son champ d'action pour lui permettre de s'incorporer dans des scènes réelles ou synthétiques, et d'interagir avec les objets dans ces scènes.

Ces opérations incluent :

1. La détermination de la visibilité entre les objets par champ de lumière, ainsi qu'avec d'autres types d'objets. Bien que ces objets n'aient a priori aucune géométrie, il est quand même nécessaire de pouvoir les "situer" par rapport aux autres, pour déterminer l'occlusion ;
2. La ré-illumination de l'objet par champ de lumière pour refléter l'illumination de la scène sur ce dernier. La ré-illumination de la scène due à l'impact de l'objet par champ de lumière est aussi un autre problème intéressant, quoique moins relié à nos travaux ;
3. La modification des propriétés de surface de l'objet. Cette opération explore la question suivante : peut-on faire l'édition d'un objet représenté par champ de lumière ? (pour ce faire, il est sous-entendu qu'on puisse être capable de récupérer les propriétés de surface) ;
4. L'animation d'objets représentés par champ de lumière. Si on ne veut pas se limiter à des objets rigides, on doit trouver une méthode pour permettre de varier la représentation par rapport au temps.

Il va sans dire que ces avenues de recherche pourraient séparément constituer des projets de recherche à part entière. À l'intérieur de cet ouvrage, nous explorerons les problèmes associés avec la ré-illumination de champ de lumière (item 2).

Aussi, nous explorerons la possibilité d'effectuer un précalcul de la visibilité grâce à des structures reliées à celles utilisées en rendu par champ de lumière, que nous appellerons "champ de visibilité". Il est important de noter que ce dernier problème est assez différent de celui énuméré à l'item 1.

1.3 Aperçu des différents chapitres

Nous étalerons d'abord les travaux antérieurs pertinents à notre travail (chapitre 2). Plus particulièrement, nous décrirons en détail les articles de Levoy et Hanrahan [LH96] et de Gortler *et al.* [GGSC96] sur lesquels notre travail est largement basé. Il va sans dire qu'une connaissance de fond de ces deux articles facilitera énormément la lecture de ce mémoire.

La présentation de notre travail sur le champ de visibilité se retrouve au chapitre 3. Notre travail sur la ré-illumination de champ de lumière sera élaboré au chapitre 4. Ces deux chapitres sont indépendants et peuvent être lus séparément. Par contre, une bonne compréhension du rendu par champ de lumière et de ses caractéristiques est nécessaire pour la lecture de ces derniers. Nous conclurons au chapitre 5.

Il est à noter que dû au caractère anglophone de la littérature et aux termes couramment utilisés dans notre domaine, bien que nous ayons fait un effort considérable pour tenter de restreindre au minimum le nombre de termes de l'anglais, il est possible que ce document soit difficile à lire à cause de la non-familiarité des termes francophones équivalents utilisés. Nous référons le lecteur au glossaire pour tenter de résoudre toute source d'ambiguïté éventuelle, où les traductions utilisées dans ce document sont énoncées.

Chapitre 2

Travaux antérieurs

Le sujet traité dans cet ouvrage étant inspiré des récents travaux en rendu basé sur des images, nous décrivons sommairement dans ce chapitre les travaux antérieurement réalisés dans ce domaine.

Ces derniers se divisent naturellement en deux catégories : d'une part, un premier ensemble de techniques considère la création de nouvelles images à partir de techniques d'interpolation entre un petit ensemble d'images de référence ; d'autre part, un autre ensemble de techniques utilise un échantillonnage plus complet et organisé du champ de lumière pour recréer des images à partir de points de vue nouveaux. Nous nous attarderons plus longuement à définir la principale technique de ce type, le rendu par champ de lumière, car ce travail est largement basé sur les idées puisées dans son développement.

2.1 Concepts généraux

En comparaison aux techniques de rendu traditionnelles, les techniques de rendu basé sur des images (*image-based rendering*) représentent un nouveau départ par rapport au but de photoréalisme[1] que l'infographie s'est donné depuis les vingt dernières années. En effet, depuis

[1] Ce terme, couramment utilisé, exprime l'intention de produire des images qui au niveau de la qualité, se comparent à des photos. On cherchera plus généralement à produire des images qui ressemblent à la réalité, voire même chercher à reproduire le stimulus dont l'individu fait l'expérience lorsqu'il "voit". C'est d'ailleurs vers ce but que plusieurs travaux de recherche récents se dirigent en effectuant un post-traitement sur des images de radiance, le *tone-mapping* [WLRP97].

les algorithmes de rendu par balayage de lignes [Gou71], en passant par le tracer de rayons [Whi80], jusqu'aux techniques de radiosité [GTGB84] et de simulation par Monte Carlo (tracer de chemins [Kaj86], tracer de rayons bidirectionnel [LW93], *photon map* [Jen95], etc.), le but d'un logiciel de rendu est de produire une image qui ressemble à la réalité. Seulement récemment a-t-on commencé à s'intéresser à produire des images avec une allure intentionnellement non réaliste (par exemple, dans le cadre de l'illustration produite par ordinateur [LS95]).

Les techniques de rendu basé sur des images ont été introduites en partie pour essayer d'éviter l'étape de modélisation, ainsi que pour tenter d'accélérer et d'améliorer le rendu en utilisant l'information contenue dans des images capturées du monde réel ou précalculées d'une scène. Elles ne visent donc pas nécessairement une application où on produit des images qui ont l'air "vraies". Essentiellement, toutes ces techniques se basent sur un échantillonnage de la lumière dans une scène soit par voie d'images, ou encore à partir d'un échantillonnage dense du champ plénoptique ou de lumière. Nous décrivons ces deux techniques aux sections 2.2 et 2.4.

2.1.1 Modélisation standard

La modélisation consiste typiquement à spécifier la forme des objets en les représentant, par exemple, par leurs limites frontières. De nombreuses techniques sont utilisées pour représenter ces objets [FvDFH90]. La création de modèles complexes et détaillés est longue et fastidieuse et on aimerait trouver un moyen de spécifier plus aisément la géométrie, et rendre la vie plus facile à l'artiste. De plus, la reconstruction "à la main" de scènes réelles est difficile à cause du grand nombre de mesures à acquérir.

Les textures à appliquer sur les objets sont aussi difficiles à créer, particulièrement si on veut simuler l'usure correcte d'un objet. De plus, les propriétés de réflectance qui varient sur les surfaces sont difficiles à spécifier en modélisation ordinaire.

Aussi, un problème relié au rendu par modèles est que la taille de ces derniers croît sans cesse, et que de nouvelles méthodes de rendu sont nécessaires pour gérer cette plus grande complexité. Pour faire des rendus à des vitesses interactives, on est jusqu'à maintenant limité

aux méthodes par balayage de lignes et la complexité de ces méthodes croît linéairement avec le nombre de polygones à rendre (i.e. la complexité de la scène).

2.1.2 Modélisation par images

La modélisation par images (*image-based modeling*) est reliée au rendu basé sur des images de par la géométrie de reprojection similaire à l'interpolation de vues. Le but de la modélisation par images est de construire un modèle géométrique d'une scène à partir de photos. Ce champ est relié de très près à la photogrammétrie, où on tente d'extraire l'information 3D (forme) d'un objet à partir de projections en perspective de ce dernier. Une application typique utilise des images aériennes dans le but de produire des cartes topographiques. Un avantage de cette méthode de modélisation réside dans le fait qu'on n'ait pas à prendre toutes les mesures nécessaires d'une scène pour pouvoir la reconstruire. Aussi, il devient possible de travailler sur la récupération des propriétés de surface des objets (voir par exemple les travaux récents de Yu et Malik [YM98]).

Les approches connues à la modélisation *automatique* par images ne donnent que des modèles incomplets et qui contiennent beaucoup d'erreurs. Quelques systèmes assistés existent, qui permettent d'effectuer de la reconstruction avec l'aide de l'usager [DTM96] [POF98]. Ces techniques peuvent fournir de meilleurs modèles, bien que plus simples, et sans propriétés réelles des surfaces.

Nous ne nous attarderons pas sur ce domaine dans notre discussion. Nous nous concentrerons plutôt sur l'aspect du rendu à partir d'images.

2.2 Rendu par interpolation d'images

Les systèmes de rendu basé sur des images créent de nouvelles vues d'une scène à partir d'un ensemble de photographies ou d'images synthétiques pré-capturées. C'est-à-dire qu'il n'y a d'autre information que ces images et possiblement leur calibration (position, orientation, etc.) qui est connue par le système. Ces nouvelles techniques de rendu offrent l'avantage que la complexité des modèles est sans limite, et le temps de rendu est indépendant de la complexité (il dépend plutôt de la taille de l'image à produire).

2.2.1 Apposition de textures et *Movie-Maps*

L'idée de base des *Movie-Maps* fut proposée par Lippman [Lip80] lors d'une application du disque vidéo (*video-disc*) dans les années 1980. Un exemple de *Movie-Maps* inclut une conduite automobile dans une ville, en réalité virtuelle. Étant donné que le disque vidéo offre une quantité de stockage très grande, il devenait possible de capturer des images de plusieurs points de vue qu'on pourrait vouloir visiter. Bien que le système était limité à l'utilisation d'une seule image à la fois depuis un point de vue fixe, une procédure de reprojection fut suggérée pour interpoler certaines transitions entre les images (une reprojection sur un plan en rotation pour simuler un virage à une intersection).

L'idée de capturer toute la lumière incidente en un point de l'espace dans une carte d'environnement [BN76] [Gre86] précède les récentes approches de rendu basé sur des images. On l'utilisait à l'origine pour effectuer des réflexions sur des objets spéculaires (e.g. chrome). On a montré que même si des projections sphériques seraient mieux appropriées, il est plus efficace d'utiliser un cube pour stocker les cartes d'environnement. Les images panoramiques (cylindriques [Che95] ou sphériques [SS97]) ont aussi été utilisées pour stocker des cartes d'environnement pour simuler l'exploration d'une scène à partir de points de vue fixes.

La technique consiste alors à sélectionner et reprojeter à la caméra une partie de l'image panoramique. Cette technique est utilisée dans le système *Quicktime VR* [Che95]. Un problème associé à cette technique est qu'on ne peut créer des images qu'à partir de points de vue fixes (i.e. pour changer la position de l'observateur, on change d'image panoramique parmi un ensemble d'images pré-calculées à diverses positions dans la scène). On peut changer la direction du point de vue, et on peut réduire la région visionnée, ce qui nous donne l'impression d'avancer dans la scène (bien entendu, plus on fait un tel "zoom" sur une zone de l'image panoramique, plus l'image est différente de celle qui serait réellement obtenue si on avançait réellement dans la scène, à cause de l'effet de perspective manquant).

2.2.2 Déformation d'images

Le *morphing* [Wol90], ou déformation d'images, est une technique qui permet de générer des vues intermédiaires d'entre deux vues connues. Une interpolation naïve de deux images ne

donne typiquement pas de bons résultats en soit, et il faut fournir au système des correspondances entre les éléments importants des deux images [BN92].

Même avec cette information, une simple interpolation linéaire des deux images crée des distorsions dans les image intermédiaires. En fournissant la calibration des images dans l'espace ainsi qu'une correspondance entre les pixels d'une image et de l'autre, il devient possible de reprojeter les images sur un plan commun et d'effectuer la déformation dans cet espace 2D, ce qui permet la génération d'une transition plus naturelle en conservant l'effet de perspective et la forme des objets [SD95] [SD96] (*shape-preserving morph*).

Pour la plupart de ces techniques, les correspondances sont fournies par l'usager de façon interactive. Bien que les artefacts associés aux erreurs des correspondances soient visuellement tolérables (e.g. *ghosting*), on obtient de meilleurs résultats en utilisant un algorithme de correspondance stéréo pour fournir au système une carte de disparité. Mais cette technique peut potentiellement créer des transitions non naturelles s'il y a des erreurs dans les correspondances fournies au système. De plus, il est difficile de traiter correctement les occlusions avec cette méthode et il est nécessaire que plusieurs formes soient facilement identifiables dans les images de référence. Cette technique représente donc un intérêt limité quant à son utilisation pour représenter de façon plus générale des scènes tridimensionnelles.

2.2.3 Interpolation de vues

La reprojection d'image (*forward mapping*) consiste à calculer la position relative d'un échantillon (pixel) dans une vue par rapport à une autre image (une image de référence) [CW93a] [LF94]. La technique appelée *interpolation de vues* consiste à calculer une vue intermédiaire complète avec cette technique. La contrainte de position fixe de la caméra est levée et donc on peut générer des vues entre deux images de référence. Ces méthodes utilisent une carte de disparité calculée par un algorithme de correspondance stéréo pour effectuer la reprojection. Cette technique requiert beaucoup de correspondances valides pour bien fonctionner. On a aussi étendu ces techniques pour permettre de combiner la reprojection de plusieurs images source, ainsi que d'utiliser des images panoramiques [MB95].

Un problème important avec cette approche est que les images originales ne donnent au-

cune information sur les régions qui sont cachées dans toutes les images. En effet, il est possible qu'une surface qui n'apparaît pas dans les images données devienne visible à partir du nouveau point de vue désiré, ou encore qu'une surface dont la projection est petite dans les images de référence devienne grande depuis le point de vue d'où on désire créer l'image, et donc que cette surface soit sous-échantillonnée (la surface apparaîtra clairsemée, avec peu de pixels pour la représenter ; ceci est le problème "d'étirement" d'une surface). On assiste alors au problème des "trous". Plusieurs techniques ont été proposées pour tenter de remplir ces trous de manière cohérente, dont diverses techniques d'interpolation. Entre autres, on peut effectuer une triangulation sur les pixels disponibles et déformer la surface triangulée en utilisant les positions des sommets reprojetés [MMB97] (pour éviter le problème d'étirement), ou encore simplement utiliser les valeurs des pixels environnants [CW93a] (pour alléger le problème des trous dus à l'occlusion). Chen [CW93a] effectue la reprojection par blocs de pixels, pour des raisons d'efficacité, mais si la reprojection est effectuée pixel par pixel, on pourrait songer à utiliser un algorithme de "splatting[2]" [SGHS98] qui dessine la contribution d'un pixel à l'image reprojetée en utilisant un noyau de taille variable, pour éviter les trous dus à l'étirement des surfaces. Mais si en général ces techniques améliorent quelque peu l'image finale, elles ne recouvrent pas les zones manquantes correctement.

2.2.4 Extensions à l'interpolation de vue

Quelques extensions à l'interpolation de vue ont été présentées récemment. Shade *et al.* [SGHS98] proposent d'utiliser des images qui contiennent en chaque pixel plusieurs valeurs de couleur et profondeur (*layered depth images*), de façon à éviter le problème de l'occlusion. La différence entre cette méthode et la simple reprojection de points 3D à l'écran est qu'ils utilisent la cohérence de l'image d'entrée avec l'algorithme proposé par McMillan et Bishop [MB95] qui permet de faire le "splatting" des pixels dans l'ordre du plus loin au plus proche, ce qui fait qu'on n'a pas besoin de tampon de profondeur. Aussi, une fois qu'on a calculé la matrice de transformation de l'image de référence à l'image désirée, on n'a pas besoin

[2] Ajout de la contribution d'un pixel par convolution d'un noyau Gaussien ou autre au point de projection du pixel sur l'image désirée. On utilise typiquement l'opérateur de composition *over* [PD84].

de faire les calculs en trois dimensions (donc moins de calculs à effectuer).

Aussi, pour tirer avantage de la cohérence des colonnes d'une image, Rademacher et Bishop [RB98] proposent de créer une image avec de multiples points de projection (*multiple center-of-projection image*, ou *MCOP*). Cette image consiste en une série de colonnes capturées à partir de points de vue le long d'une trajectoire de caméra associée avec l'image.

2.3 Fonction plénoptique

Dans les sections suivantes, le concept d'image sera généralisé à l'espace, de façon à caractériser les images nécessaires pour reconstruire des vues arbitraires d'une scène. On échantillonnera de façon dense la radiance autour d'un objet. Ce champ de radiance (mesuré en *Watts*$/m^2 \cdot sr$) est appelé fonction plénoptique[3] [AB91].

Cette fonction peut être décrite comme la radiance reçue en un point (x, y, z) de l'espace, depuis une direction (θ, ϕ) par rapport à ce point, à un moment t, pour une longueur d'onde donnée λ. On peut donc exprimer cette fonction sous la forme

$$P(x, y, z, \theta, \phi, \lambda, t).$$

De plus, on pourra ajouter la cohérence (propriété de la phase de l'onde) et la polarisation (orientation autour de l'axe de propagation) de l'onde incidente si nécessaire [CW93b].

Dans le cadre de notre travail, ainsi que pour la plupart des travaux antérieurs en rendu par champ de lumière, on néglige les paramètres suivants :

- **Temps** t : on ne considère que des scènes statiques, et où on suppose la propagation instantanée (vitesse infinie) de la lumière en régime permanent ;
- **Longueur d'onde** λ : on traite séparément trois longueurs d'onde principales, correspondant aux trois longueurs d'onde associées aux phosphores d'un écran cathodique (i.e. canaux R, G et B) ;
- **Phase** : ce paramètre a en général peu d'effet sur l'image finale ;

[3] du latin *plenus*, plein et *optic*, ce qui a trait à la vision [MB95].

- **Cohérence** : ce paramètre a en général peu d'effet sur l'image finale.

On se retrouve alors avec une fonction de cinq dimensions $P(x, y, z, \theta, \phi)$ qui exprime la radiance en un point donné qui se propage dans une direction donnée (unidirectionnelle, donc dans un seul sens).

Dans le vacuum, la radiance est une quantité qui est constante dans sa direction de propagation. Cet axiome nous permet de réduire la dimension du champ plénoptique à quatre, c'est-à-dire au domaine de toutes les lignes dans l'espace. Nous appellerons cette fonction *champ de lumière*. Différentes paramétrisations du domaine des lignes dans l'espace pour représenter ce champ seront présentées à la section 2.4.1.

2.4 Rendu par champ de lumière ou *lumigraph*

Une approche différente de l'interpolation d'images utilise un échantillonnage systématique du champ de lumière. Cette approche fut présentée simultanément par Levoy et Hanrahan [LH96] (*light field rendering*) et Gortler *et al.* [GGSC96] (*lumigraph*). Dans le cadre de cet ouvrage, nous l'appellerons *rendu par champ de lumière*.

L'idée intuitive de cette technique est la suivante : pour reproduire des images d'un objet, il suffit de connaître le champ de lumière qu'il produit. Si on peut mesurer cette fonction (le champ de lumière) et la stocker, il suffit de la ré-échantillonner pour obtenir une approximation des valeurs de radiance désirées pour créer une nouvelle image de l'objet (celles dirigées vers la caméra depuis laquelle on veut créer une image).

Cette technique s'effectue en trois étapes :

1. Établir une paramétrisation discrète d'un ensemble de lignes dans l'espace ;
2. Échantillonner le champ de lumière et construire une représentation de ce champ sur la paramétrisation établie en (1) ;
3. Faire le rendu d'images en ré-échantillonnant ces données à partir d'un nouveau point de vue.

Chacun de ces aspects seront élaborés dans les sous-sections qui suivent. Un diagramme du processus général est illustré à la figure 2.1.

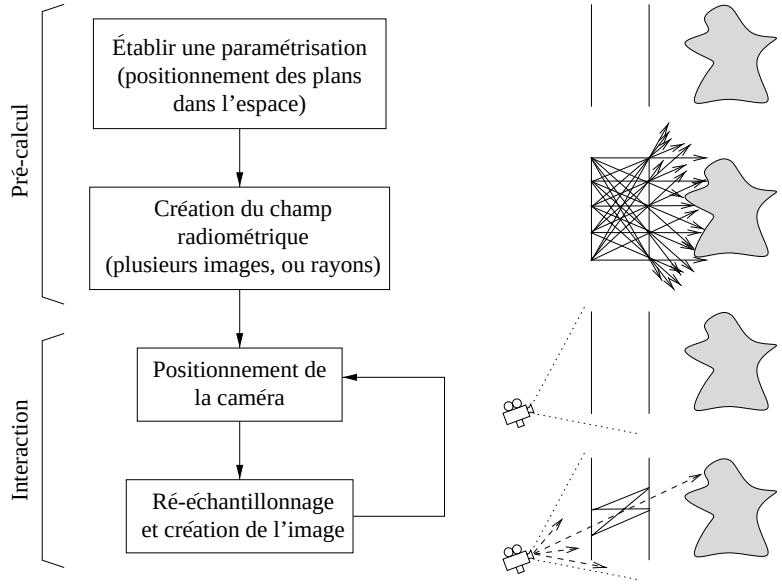


FIG. 2.1: Processus du rendu par champ de lumière.

2.4.1 Paramétrisation des lignes dans l'espace

Une des conséquences importantes de la simplification à quatre dimensions de la fonction plénoptique discutée en section 2.3 est que cette technique ne s'applique que dans l'une des deux configurations suivantes :

- Pour un objet isolé dans une région convexe qu'on regarde depuis l'extérieur ;
- Pour une région convexe vide à l'intérieur de laquelle on regarde vers l'extérieur.

On doit donc pouvoir établir une paramétrisation des lignes pour un espace fermé convexe (i.e. une sphère topologique).

Paramétrisation à deux plans (2PP)

L'utilisation d'une paramétrisation à deux plans est proposée par Levoy et Hanrahan [LH96] et Gortler *et al.* [GGSC96]. Elle consiste à positionner deux rectangles planaires dans l'espace et de considérer les coordonnées d'intersection d'une droite avec ces deux rectangles[4]. Cette paire d'intersections est unique. On discrétise le domaine en discrétisant chacun

[4]Il est important de noter que nous utiliserons dès lors le terme "plan" en référence aux rectangles de la paramétrisation, et non pour parler d'un plan infini.

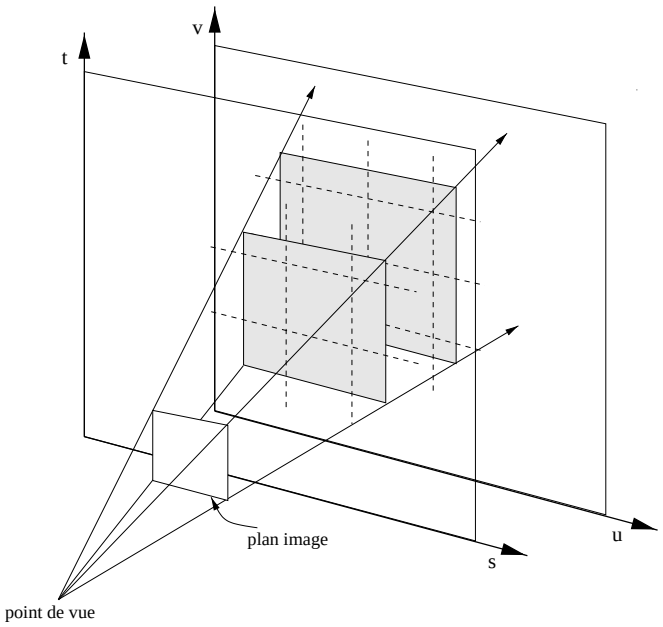


FIG. 2.2: Ré-échantillonnage de la paramétrisation à deux plans.

des plans. Ces derniers sont, en pratique, des rectangles finis incidents avec les plans.

La caractérisation des lignes choisies avec cette paramétrisation est étudiée par Levoy et Hanrahan, et encore plus en détail par Gu *et al.* [GGC97]. On cherche typiquement un ensemble de lignes pour lesquelles on aura une résolution spatialement uniforme *et* directionnellement uniforme.

Levoy et Hanrahan [LH96] font une brève analyse de la qualité des 2PPs en exprimant une version bidimensionnelle de ce type de paramétrisation sous forme de coordonnées polaires, ce qui permet de juger de l'étendue à la fois spatiale et directionnelle des lignes choisies pour l'échantillonnage. On peut alors observer qu'un bon choix pour une 2PP consiste à positionner le plan uv à l'infini (derrière les positions potentielles des caméras associées aux points de vue désirés) de taille très grande, et le plan st près de l'objet (typiquement devant, mais ça ne change rien), d'une taille similaire à l'objet. Avec un tel positionnement, si on fixe un point sur le plan st, on obtient une image cisaillée, alors que si on fixe un point sur le plan uv assez près d'un objet, on obtient une image qui ressemble à une BRDF échantillonnée de la surface au point (voir figure 4.16 à la page 76). Une série d'images uv à partir de st fixes sont illustrées à la figure 2.3 pour le champ de lumière d'un noeud mathématique. Une série d'images st à

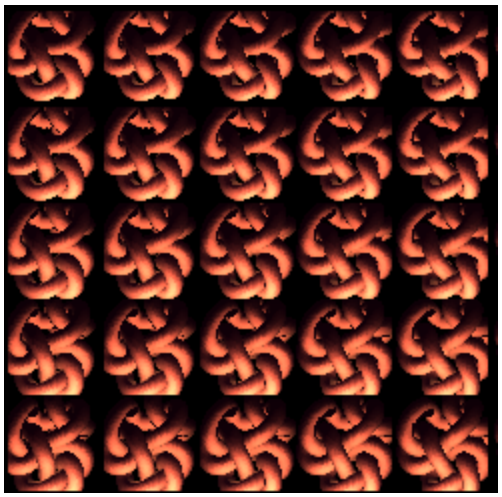


FIG. 2.3: Série de 25 images st à partir de uv fixes.

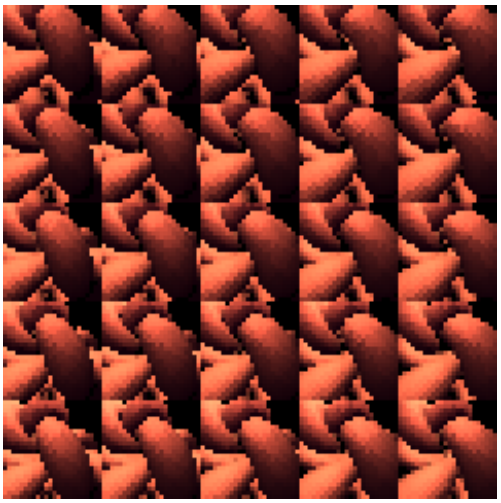


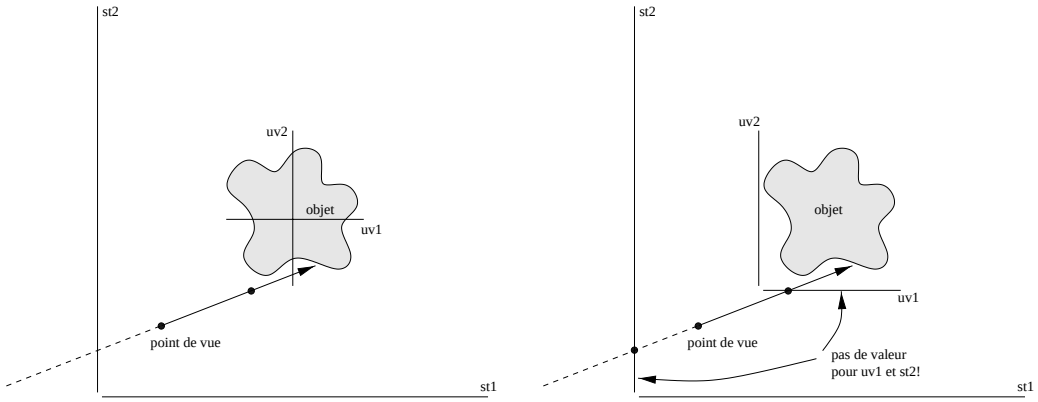
FIG. 2.4: Série de 25 images uv à partir de st fixes.

partir de uv fixes sont illustrées à la figure 2.4.

Un des avantages de la 2PP est son efficacité. Étant donné un rayon pour lequel on désire obtenir la radiance (comme ceux qu'on lancera depuis le point de vue, un *query ray*), on peut rapidement obtenir les coordonnées d'intersection du rayon avec les plans (quelques multiplications et additions, aucune opération trigonométrique). Cette méthode est à la fois plus simple et moins coûteuse que l'utilisation des coordonnées de Plücker [Som59] pour représenter des lignes dans l'espace.

La 2PP permet seulement de couvrir un sous-ensemble de lignes (i.e. celles qui passent par les deux plans). Pour représenter un objet au complet, on a besoin de représenter les lignes tout autour de ce dernier. Pour ce faire, on peut utiliser plusieurs 2PPs. Il faut alors faire passer les plans uv par le centre de l'objet (voir figure 2.5(a)). Intuitivement, on est d'abord porté à croire qu'en plaçant les plans uv des paramétrisations "autour" de l'objet on couvrira tout le domaine, mais c'est une erreur. Un exemple d'un rayon qui n'est pas couvert par cette configuration est démontré à la figure 2.5(b).

L'utilisation de multiples 2PPs n'est pas sans problèmes : la distribution des échantillons lorsqu'on ré-échantillonne une 2PP depuis un point de vue dont la direction est parallèle avec les plans n'est pas très dense. Il y a aussi des artefacts à la limite entre deux 2PPs, et il est difficile d'appliquer des filtres de ré-échantillonnage à cette limite. Par exemple, lorsqu'on a



(a) Correct (b) Incorrect

FIG. 2.5: Positionnement de plusieurs 2PPs.

une configuration de deux paires de paramétrisations perpendiculaires, si on place la caméra de telle façon à ce qu'on doive utiliser les deux paramétrisations pour couvrir l'image voulue, la densité d'échantillons pour chacune des zones de l'image correspondantes sera assez différente.

Dans notre travail, seulement la paramétrisation à deux plans sera considérée, pour sa simplicité et son efficacité. De plus, nous utiliserons seulement une seule 2PP à la fois dans le cadre de nos expériences (ceci ne change rien aux résultats et n'est pas une limitation des techniques que nous présenterons).

Paramétrisation à deux sphères (2SP)

Pour tenter de pallier à ces problèmes, une paramétrisation sur la sphère a été développée [CLF98], qui est plus uniforme dans le domaine des lignes en 3D. Avec cette paramétrisation, la qualité du rendu ne dépend plus de l'orientation de la caméra relativement à la paramétrisation.

La paramétrisation à deux sphères consiste à distribuer uniformément des points sur la sphère, et à définir un échantillon pour chaque paire de points sur cette sphère. Chacune de ces paires de points définit une ligne dans l'espace, ligne qui correspond à la direction et position de la radiance échantillonnée à cet endroit. On choisit typiquement une sphère qui englobe l'objet considéré, de façon à pouvoir l'observer depuis tous les points de vue possibles. Ceci nous permet d'éviter les problèmes dus à l'utilisation de plusieurs paramétrisations pour représenter

un seul objet. Pour implémenter rapidement cet algorithme, la sphère est triangulée en triangles de tailles identiques et un algorithme de subdivision adaptative est appliqué sur chacun des triangles pour stocker les échantillons à partir de points sur les triangles.

Même si cet algorithme donne de meilleurs résultats au niveau ré-échantillonnage, il est plus coûteux et complexe. Un des désavantages importants de cette technique est qu'on ne puisse pas exploiter la cohérence d'un algorithme par balayage de lignes lors de la construction et du ré-échantillonnage, comme on peut aisément le faire avec une 2PP en utilisant des calculs incrémentaux.

Paramétrisation plan-sphère (SPP)

La paramétrisation plan-sphère aussi proposée par Camahort *et al.* [CLF98] tente de faire un compromis entre la 2SP et la 2PP, en essayant d'exploiter la cohérence des lignes de balayage tout en conservant une uniformité d'échantillonnage tout autour de l'objet. Elle utilise un point choisi uniformément sur la sphère pour spécifier la direction de la ligne, ainsi qu'une position sur le plan projeté au centre de la sphère pour spécifier la position spatiale de la ligne. Cette paramétrisation affiche certains artefacts similaires à ceux de la 2PP, et est moins satisfaisante que la 2SP.

2.4.2 Échantillonnage du champ de lumière

Une fois la paramétrisation établie, il faut échantillonner le champ de lumière pour le stocker. Un problème inhérent à la création de champs de lumière repose dans l'illumination qui est aussi capturée à ce moment (voir au chapitre 4 pour une discussion détaillée de ce problème).

Préfiltrage

Comme on construit une représentation discrète du champ de lumière, il est indiqué d'effectuer un préfiltrage pour améliorer la qualité d'approximation de chaque échantillon de radiance. Dans le domaine des lignes dans l'espace 3D, ceci se traduit par le fait que chaque échantillon discret représente un sous-ensemble de lignes avoisinantes (plus ou moins,

dépendant de la résolution sur les plans de paramétrisation). On utilisera un filtre de reconstruction invariant selon la position (il n'est pas évident de spécifier un filtre de reconstruction qui soit adapté à la distribution directionnelle variante de chaque échantillon. Il serait néanmoins intéressant d'étudier plus en profondeur les caractérisques de ce processus de filtrage).

Scène synthétique

Une technique simple pour créer un champ de lumière consiste à effectuer un ou plusieurs rendus (tracers de rayons) sur une scène synthétique. On crée une image pour chaque s, t fixe, avec la position de la caméra au point s, t choisi, et avec une projection oblique qui englobe précisément le plan uv. Pour préfilter, les techniques classiques de suréchantillonnage peuvent être utilisées [Gla89]. On doit effectuer le préfiltrage en considérant à la fois la nature discrète de uv ainsi que de st (on filtre sur les quatre paramètres[5]). Aussi, un avantage à utiliser des scènes synthétiques est qu'on peut choisir de stocker seulement la réflexion due à une illumination ambiante (c'est-à-dire qui se propage partout, dans toutes les directions avec la même intensité[6]) avec des surfaces complètement diffuses, ou encore stocker les propriétés de surface des matériaux au lieu de conserver le résultat de la réflexion (la couleur, voir section 4.7.9).

On peut aussi utiliser n'importe quel moteur de rendu pour créer les échantillons, en spécifiant une projection oblique où le point de vue se situe sur le plan st et le plan image est incident avec le plan uv. C'est ce que font, par exemple, Levoy et Hanrahan [LH96].

Scènes réelles

Pour capturer le champ de lumière de scènes réelles, on utilise des caméras pour obtenir plusieurs images de la scène.

Levoy et Hanrahan [LH96] utilisent une caméra calibrée montée sur un support mécanique qui prend des images d'un objet à intervalles réguliers correspondant à la discrétisation sur un des plans de la 2PP. Pour préfiltrer en uv, ils utilisent une caméra rotative, ce qui simule l'intégration des valeurs pour un pixel sur uv. Pour préfilter en st, ils suggèrent de simuler

[5] Si on considérait le rendu de scènes avec des objets en mouvement, on devrait aussi préfiltrer en considérant la variable temps.

[6] Ce genre d'approximation est souvent nécessaire dans le cadre du tracer de rayons.

l'ouverture de diaphragme de façon synthétique en faisant la moyenne de quelques images prises à partir de points de vue adjacents.

Gortler *et al.* [GGSC96] utilisent un mécanisme plus complexe : ils prennent plusieurs images aléatoirement autour de l'objet considéré (une caméra vidéo portative est utilisée pour balayer le champ) et construisent par la suite les échantillons pour la paramétrisation par un processus de refonte (*rebinning*) coûteux. Pour calibrer la caméra, ils placent l'objet devant un arrière-plan soigneusement mesuré et constitué d'un motif qu'on peut détecter de façon automatique.

2.4.3 Rendu

Le rendu est effectué en ré-échantillonnant le champ de lumière depuis le point de vue désiré. On considère l'intersection d'un rayon qui passe par la position de l'oeil et d'un pixel de l'image à créer, et qui intersecte les deux plans de la paramétrisation.

Postfiltrage

Lors du rendu, la reconstruction de la valeur d'illumination désirée pour un pixel sera très importante (surtout pour certains points de vue, par exemple, si la caméra est très proche de l'un des deux plans). Il est nécessaire d'effectuer une reconstruction de la valeur de radiance désirée à partir des échantillons avoisinant les coordonnées d'intersection d'un rayon provenant de la caméra. Si les rayons de la paramétrisation sont plus espacés que ceux de la nouvelle image, ceci introduira du flou dans l'image. C'est un problème classique de reconstruction de signal.

Rendu par balayage de lignes

Pour accélérer le rendu, Levoy et Hanrahan [LH96] exploitent la cohérence de la projection des plans sur le plan image en faisant une conversion par balayage de lignes de l'intersection des plans de la paramétrisation. Ensuite, chaque pixel se voit assigné une valeur correspondant soit à l'échantillon (u, v, s, t) le plus près des points d'intersection réels du rayon à travers le pixel, soit à une combinaison des quatre échantillons les plus près en uv ou des quatre échantillons les plus près en st (par une interpolation bilinéaire), ou soit encore à une combi-

naison des seize échantillons les plus près en uv *et* st (ce qui revient à faire une interpolation quadrilinéaire). Comme la conversion est effectuée sur des surfaces planaires, tous les calculs de coordonnées peuvent s'effectuer de façon incrémentale (très rapide).

Rendu par apposition de textures

La technique utilisée par Gortler *et al.* [GGSC96] utilise, quant à elle, l'implémentation matérielle de la conversion par balayage de lignes des polygones et l'apposition de textures maintenant couramment fournies par les cartes graphiques 3D avancées. Il est possible de discrétiser le plan image en régions (triangles) pour lesquelles les valeurs de pixels seront puisées depuis une paire st fixe, et de faire une apposition de texture d'une portion de l'image uv correspondant à ce st sur cette région.

Pour effectuer le postfiltrage dans ce contexte, ils utilisent la fusion alpha du matériel, c'est-à-dire qu'au lieu d'afficher un seul polygone texturé de l'image uv, ils rendent successivement les quatre polygones des images uv correspondant aux coordonnées st adjacentes à la région considérée. Le matériel supporte directement l'interpolation bilinéaire des texels, qu'on utilise ici pour interpoler sur le plan uv. On peut aussi tenir compte de l'interpolation par rapport au plan st en choisissant des poids appropriés pour les canaux alpha des coins des polygones et en faisant le rendu de ceux-ci avec l'interpolation matérielle sur le polygone, de telle façon à ce que chaque pixel rendu apporte une contribution qui correspond à une interpolation bilinéaire en st (on peut subdiviser la région en quelques triangles pour mieux approximer le filtre bilinéaire). En additionnant dans le tampon image les rendus successifs de ces polygones texturés correspondant aux quatre échantillons sur le plan uv, on obtient correctement une somme qui correspond à une interpolation quadrilinéaire, équivalente à celle décrite plus haut dans cette section.

Cette méthode permet de faire plusieurs autres optimisations de vitesse intéressantes, entre autres, d'ajouter des contraintes temps-réel [SCG97]. Des triangulations de plus basses résolutions (moins coûteuses) du plan image peuvent être utilisées, introduisant ainsi un compromis entre la qualité du rendu et la vitesse (le choix des triangles utilisés influencera la qualité). Aussi, si on a un budget de mémoire limité, on peut utiliser les textures projectives

matérielles [SKv+92] pour déformer sur une approximation de la géométrie un rendu effectué précedemment dans le temps, comme Chen et Williams le font avec l'interpolation de vues [CW93a].

2.4.4 Discussion

Le rendu par champ de lumière offre l'avantage que le rendu prend alors un temps qui dépend de la taille de l'image, plutôt que de la taille ou complexité de la scène (c'est-à-dire le nombre d'objets, leur niveau de détails, les types d'objets, les types de surfaces, les sources d'illumination présentes, etc.). Aussi, il permet d'afficher des effets arbitrairement complexes, pour autant qu'on soit en mesure de les capturer.

Par contre, les aléas de la reconstruction discrète sont difficiles à surmonter. La résolution fixe de la structure de données nous empêche d'échantillonner adaptativement les zones avec des discontinuités, et le flou introduit pour alléger l'aliassage est inévitable. De plus, pour obtenir un rendu décent, la structure de données en quatre dimensions créée requiert des résolutions typiques qui engendrent des tailles prohibitives (de plusieurs megaoctets à quelques gigaoctets). Il est donc impératif de prévoir une méthode de compression qui permette d'accéder directement aux données compressées (i.e. sans avoir à décompresser le tout en mémoire). À cet effet, Levoy et Hanrahan [LH96] utilisent une combinaison de compressions de type *vector-quantization* (VQ) pour encoder en mémoire et Lempel-Ziv (*gnuzip*) pour compresser encore plus pour le stockage sur disque. Ils obtiennent des ratios de compression d'environ 40 :1 à 200 :1 dont la contribution de l'algorithme Lempel-Ziv représente environ 5:1 à 10:1. L'encodage VQ utilise de simples tables d'indirection pour accéder directement les blocs de données et est donc très efficace.

Une autre technique fondamentalement assez proche du rendu par champ de lumière fut proposée par Lewis *et al.* [LF96] (le moteur de rendu *Lucifer*). Cette technique de rendu propose de résoudre l'équation du rendu [Kaj86] en propageant itérativement la radiance sur les murs de voxels d'une scène ainsi subdivisée. Un avantage de cette technique est que si l'on peut déterminer simplement l'interaction de la lumière qui entre par les murs d'un voxel avec les objets qu'il contient, et d'en extraire la lumière sortant sur ces mêmes murs (réfléchie ou

réfractée par les objets dans le voxel, ou encore directement propagée à travers ce dernier), il suffit d'appliquer itérativement un algorithme de transfert de la lumière entre les voxels (la lumière qui sort d'un mur de voxel devient la lumière qui entre sur son voisin de l'autre côté du mur). Pour arriver à stocker toute cette information (radiance en fonction de la position i, j et de la direction θ, ϕ, sur *chacun* des murs des voxels, chaque mur contenant une représentation en quatre dimensions), ils proposent une représentation compacte et hiérarchique sous forme d'ondelettes pour chacun des murs des voxels.

Éventail des intensités utilisées pour stocker la radiance

Dans la plupart des travaux antérieurs en rendu basé sur des images, les valeurs de radiance sont simplement stockées en trois canaux RGB, sur deux ordres de magnitude ($[0, 256[$). Pour capturer les valeurs de radiance observées dans la nature il faut environ neuf ordres de magnitude [WL97b]. Il serait donc important d'utiliser une technique de stockage qui permette de stocker cet intervalle de valeurs (e.g. LogLUV [WL97a]).

2.5 Rendu basé sur les vues

Il existe une autre technique située entre le rendu par champ de lumière et le rendu classique (i.e. à partir d'un modèle) : le rendu basé sur les vues [PCD+97]. Cette technique utilise le fait qu'on n'ait besoin que de la géométrie visible pour effectuer le rendu. On précalcule et stocke des maillages partiels texturés et seulement ceux qui sont pertinents à un point de vue donné sont considérés pour le rendu. Une esquisse de l'algorithme est illustrée à la figure 2.6.

La fusion alpha est effectuée en considérant plusieurs paramètres (proximité des silhouettes, densité d'échantillonnage de surface, position relative de la caméra par rapport aux trois points de vue de référence, etc.) pour atténuer les discontinuités dues aux changements de visibilité (ils nomment cette technique *soft z-buffering*). Toutes ces opérations peuvent être effectuées avec support matériel, donc très rapidement. Par contre, il y a beaucoup de redondance dans l'information conservée (les maillages se recouvrent), et on doit capturer la forme de l'objet (image de profondeur) pour créer ces structures, ce qui n'est pas aisément réalisé et non-nécessaire pour le rendu par champ de lumière.

En précalcul
begin
- Capturer un nombre de maillages partiels (à partir d'images de profondeur) depuis plusieurs points de vue tout autour de l'objet considéré
- Trianguler une sphère englobant l'objet en utilisant comme sommets les points de vue utilisés pour capturer les maillages partiels à l'étape précédente. Créer et conserver des versions à basse résolution de chacun de ces maillages avec leurs textures

end

Au rendu
begin
- Projeter la position de l'oeil sur la sphère pour trouver les trois points de vue les plus près
- Effectuer un rendu classique (par balayage de lignes) des trois maillages correspondant à ces trois points de vue
- Faire la fusion alpha de ces trois rendus

end

FIG. 2.6: Algorithme du rendu basé sur des vues.

Chapitre 3

Champ de visibilité

Dave : Where the hell did you get that idea, HAL ?
HAL : Dave, although you took thorough precautions in the pod against my hearing you, I could see your lips move.

—2001 : A Space Odyssey

Nous traiterons dans ce chapitre du calcul de visibilité à l'aide de concepts reliés au rendu par champ de lumière. Plus précisément, nous utiliserons la paramétrisation à deux plans pour tenter d'échantillonner la visibilité.

Cette excursion est fondée sur l'idée que le calcul de visibilité en tracer de rayons représente la majorité du coût de cette technique. Le précalcul de visibilité apporte donc une accélération majeure à cette technique, ainsi que pour les techniques de balayage de lignes (si on a moins de primitives géométriques à discrétiser).

Nous évoquerons d'abord les travaux antérieurs dans le domaine de la visibilité, et décrirons ensuite notre technique d'échantillonnage ainsi que le nouvel algorithme de rendu correspondant. Les résultats de l'application de cet algorithme seront ensuite présentés et accompagnés d'une discussion.

3.1 Travaux antérieurs en calcul de visibilité

La détermination de la visibilité des surfaces en infographie représente un des plus importants sujets de recherche depuis le tout début de l'expansion rapide de ce domaine vers les

années 1970. De nombreux algorithmes d'élimination des surfaces cachées ont été proposés et sont couramment utilisés de nos jours. Plusieurs types d'algorithmes ont été proposés à cet effet :

- Détermination de la visibilité pour les dessins de lignes [Rob65] [App68] ;
- Tampon de profondeur [Cat74] ;
- Listes de priorité [FKN80] ;
- Algorithmes par ligne de balayage [BK70] [Bou70] ;
- Subdivision de l'aire (en espace image) [War69] [WA77] ;
- Tracer de rayons [Whi80] (volumes englobants).

Un traitement en profondeur de toutes ces techniques ne sera pas inclus ici. Plus de détails se retrouvent dans les articles de revue de Sutherland *et al.* [SSS74] et Gharachorloo *et al.* [GGSS89].

Même en considérant la multitude d'algorithmes existant pour calculer la visibilité, dans le contexte où la puissance sans cesse croissante des ordinateurs nous permet de travailler avec de plus en plus grosses bases de données de modèles, des algorithmes plus complexes et efficaces deviennent rapidement nécessaires pour réduire le coût du calcul de visibilité pour le rendu. C'est dans cette direction que les récents efforts de recherche se dirigent, c'est-à-dire qu'on essaie de traiter la visibilité pour un très grand nombre de polygones[1].

Bien que la solution optimale au problème de visibilité consiste à trouver toutes les surfaces visibles et seulement celles-ci, plusieurs de ces algorithmes ne fournissent que des approximations à ce problème. Dans ce contexte, il est important de bien caractériser les différentes solutions de visibilité possibles. D'une part, il faut distinguer entre une solution de visibilité *exacte*, c'est-à-dire qui retourne seulement les surfaces visibles, possiblement découpées pour en enlever les portions cachées, et une solution de visibilité *conservatrice* qui peut contenir des polygones partiellement ou même complètement cachés. D'autre part, il faut distinguer entre une solution de visibilité *complète*, pour laquelle il est garanti que toute les surfaces visibles

[1] Le problème typique consiste en la détermination de la visibilité pour une liste inorganisée de polygones, typiquement appelée *polygon soup*.

	Complète	Incomplète ou approximative
Exacte	(Squelette de visibilité)	Z-buffer (Tampon de profondeur)
Conservatrice	PVS (Ensemble potentiellement visible)	HOM (Image hiérarchique de l'occlusion) Champ de visibilité

FIG. 3.1: Caractérisations possibles d'une solution de visibilité.

auront été considérées et une solution de visibilité *incomplète* ou *approximative*, pour laquelle il est possible que des surfaces aient été négligées et soient manquantes dans la solution retournée (possiblement en offrant une garantie sur la taille maximale de ces surfaces). Ces caractéristiques sont illustrées dans le diagramme à la figure 3.1. Par exemple, l'algorithme de tampon de profondeur couramment implémenté en matériel est un algorithme exact et incomplet car aucune surface cachée ne sera retournée et la détermination de la plus proche surface visible est effectuée par échantillonnage, donc possiblement avec des erreurs à l'intérieur d'un même pixel (sauf au centre du pixel).

Plusieurs techniques éliminent les surfaces cachées à partir de structures en espace image. On peut considérer le tampon de profondeur [Cat74] comme un échantillonnage de la visibilité à partir de chaque pixel. Une méthode plus récente réutilise cette idée pour accélérer le rendu en créant au moment du rendu une version hiérarchique de ces "images de visibilité" pour éliminer de façon efficace des agrégats de polygones importants [ZMHI97]. Ils sélectionnent d'abord quelques objets susceptibles d'en cacher beaucoup d'autres et créent rapidement une image d'occlusion hiérarchique (*HOM*[2]) en effectuant le rendu de ces quelques objets. Ils utilisent ensuite cette image pour éviter de considérer des groupes entiers d'objets au rendu en vérifiant si la projection de leur boîte englobante projette dans une région de l'image cachée. De la même façon que le tampon de profondeur, comme cette technique ne stocke que la profondeur en chaque pixel, elle ne fournit pas une solution de visibilité complète (il peut y avoir des

[2]De l'anglais : *Hierarchical Occlusion Map*.

erreurs, e.g. de petits objets manquants).

Des caractérisations plus précises de la fonction de visibilité pour des scènes composées de polygones ont été développées récemment [GM90] [DDP96]. La fonction de visibilité comporte plusieurs discontinuités. Il est très important de détecter ces discontinuités, en particulier pour les calculs de radiosité par éléments finis [Hec92] [LTG92]. Plusieurs méthodes ont été développées pour tenter de détecter où apparaissent ces discontinuités (avec des contraintes (*portals*) [Tel92] ou pour calculer le graphe d'aspect [GM90]). Une méthode efficace pour effectuer ce calcul de façon globale pour une scène composée de polygones a été proposé par Durand *et al.* [DDP97]. Leur méthode permet de calculer la visibilité exacte depuis n'importe quel point dans l'espace. Dans une phase de précalcul, ils créent un graphe qui représente les événements de visibilité pour toute la scène et décrivent une méthode pour en extraire efficacement l'information de visibilité. Leur graphe peut aussi être calculé de façon paresseuse[3].

D'autres algorithmes utilisent des relations de visibilité précises entre les bloqueurs et les autres objets pour améliorer la phase d'élimination de polygones [CT97] [COFHZ98]. Comme les *HOMs*, ces algorithmes identifient les objets bloqueurs importants et vérifient la position relative de la caméra et d'un objet candidat au rendu avec ces bloqueurs pour pouvoir éliminer plusieurs polygones. Ils utilisent une approche analytique pour obtenir une estimation conservatrice de la visibilité d'un polygone.

L'échantillonnage a déjà été considéré pour calculer la visibilité volume-à-polygone [TH93]. Dans le cadre de leur système de parcours architecturaux, Airey *et al.* [ARB90] utilisent un échantillonnage pour estimer l'ensemble de polygones potentiellement visibles à partir de salles dans un bâtiment. Leur technique suppose un environnement où les polygones ont des tailles assez grandes par rapport à la résolution de l'échantillonnage.

3.2 Échantillonnage de la visibilité

La technique que nous nous sommes proposés d'explorer ici consiste en l'échantillonnage de la visibilité depuis plusieurs points de vue. La méthode présentée utilise une paramétrisation

[3] En anglais : par *lazy evaluation*.

à deux plans (2PP) pour stocker des index à des blocs de géométrie visibles dans la direction des rayons de la paramétrisation. On détermine les surfaces visibles pour un échantillon (une entrée dans la table à quatre dimensions $uvst$) par lancer de rayons et on crée une structure similaire à celle utilisée pour le rendu par champ de lumière. On utilise ensuite cette structure pour déterminer efficacement les surfaces visibles dans le cadre d'un algorithme de rendu classique.

3.2.1 Visibilité selon un rayon

Il a été démontré que la fonction plénoptique se réduit à quatre dimensions dans l'espace libre d'objets bloqueurs (voir section 2.3). Cette réduction est possible car on peut démontrer que la radiance est constante sur une ligne dans l'espace.

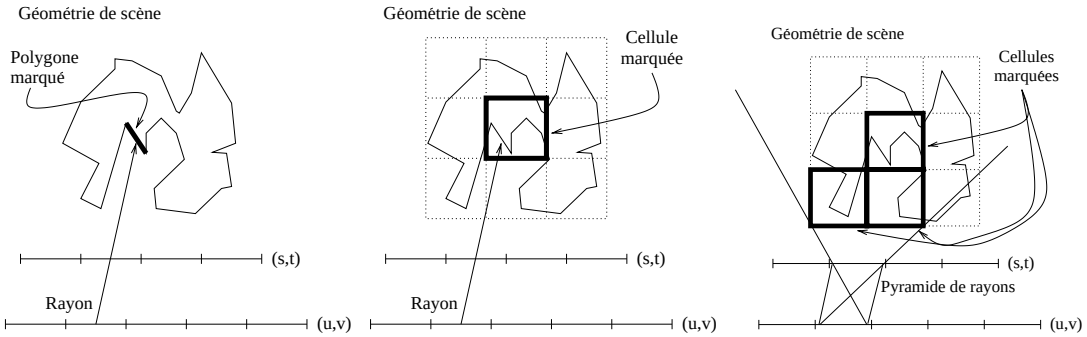
Le même principe s'applique dans le cas de la visibilité selon une ligne : la surface d'un objet qui est visible dans la direction d'une ligne est aussi constante. On peut donc représenter un "champ de visibilité selon une ligne" avec une fonction quadridimensionnelle, et les paramétrisations développées dans le cadre du rendu par champ de lumière s'appliquent ici.

Il faut encore définir comment on représentera la "visibilité selon un rayon". On fait l'hypothèse que la scène est composée d'un ensemble inorganisé de polygones[4]. On choisit alors d'assigner à un rayon donné un élément de géométrie, soit

1. Le premier polygone que le rayon intersecte ;
2. Un groupe de polygones (une cellule) qui contient le premier polygone que le rayon intersecte ;
3. Plusieurs groupes de polygones (cellules) qui contiennent le premier polygone que le rayon intersecte.

Chacun de ces choix a des avantages et des inconvénients. La première méthode consiste à associer un index à chaque polygone, et de conserver un tel index pour chaque échantillon. Comme les polygones sont typiquement petits, cette technique réduira la quantité de polygones référencés de façon redondante par plusieurs échantillons, car peu de rayons vont intersecter

[4]On parle ici de polygones bien qu'en réalité, la technique s'applique à toute forme d'objet.



(a) Un index de polygone par échantillon

(b) Un index de cellule par échantillon

(c) Plusieurs index de cellules par échantillon

FIG. 3.2: Échantillonnage de visibilité pour un uv et st.

le polygone (voir figure 3.2(a)). On voudrait éviter les références multiples redondantes, car même si elles ne causeront pas d'erreur avec l'algorithme de rendu proposé, le temps de rendu sera plus long dû au parcours de la structure de marquage (voir section 3.3). Idéalement, on voudrait obtenir une granularité pour laquelle chaque polygone *visible* sera accédé une seule fois. Aussi, comme on a beaucoup de polygones distincts à référencer, chaque index de polygone devra contenir assez de bits pour pouvoir stocker de grands nombres, ce qui pourrait accroître la taille du champ. De plus, pour des régions détaillées, les polygones peuvent être assez petits, ce qui veut dire que pour échantillonner correctement ces régions, on devra utiliser une fréquence d'échantillonnage élevée (une plus haute résolution du champ de visibilité) pour couvrir les trous (voir section 3.5.2).

La deuxième alternative consiste à grouper ensemble les polygones dans les cellules d'une grille 3D pour chaque objet (voir figure 3.2(b)). Cette méthode a pour effet d'accroître la granularité de l'échantillon de visibilité, ce qui amenuise le problème des trous. Par contre, un désavantage repose dans le fait que la taille des cellules dépend de la position des objets dans la scène (i.e. la taille des cellules projetées peut être très petite). Il faut aussi noter qu'avec cette méthode, un polygone qui se retrouve dans plus d'une cellule sera ajouté aux deux listes de polygones des cellules.

La troisième alternative consiste aussi à grouper les polygones en cellules, mais cette fois en conservant plusieurs index de cellule pour chaque échantillon (voir figure 3.2(c)). On stocke

les cellules qui intersectent les pyramides de rayons que représente un échantillon. Un problème avec cette approche réside dans le fait que la taille résultante d'un échantillon est variable (le nombre de cellules qui intersectent une pyramide est variable). Aussi, il est difficile de trouver une borne raisonnable pour la taille d'un échantillon, dans le but d'utiliser une taille fixe par échantillon. Une telle borne dépendrait de la géométrie de la scène, et devrait être calculée avant ou encore au moment de la création du champ de visibilité en calculant l'intersection des pyramides sous-tendues par chaque $uvst$ avec les cellules de la scène. Si les plans uv et st sont parallèles et rectangulaires, on pourrait envisager ce processus par un algorithme par balayage de lignes (autrement il faudrait utiliser une technique analytique d'intersection des pyramides obliques plus coûteuse).

On fait un compromis entre la quantité de géométrie qu'un échantillon de la paramétrisation référence et la qualité de l'estimation de la visibilité obtenue. Si la géométrie est divisée en grosses cellules, on pourrait référencer à plus d'objets qu'il n'est nécessaire. Par contre, si on utilise de petites cellules, l'échantillonnage de visibilité pourrait en rater certaines (tout dépendant de la méthode qu'on utilise pour créer le champ de visibilité).

Nous avons choisi d'implémenter un compromis entre la seconde et la troisième technique : on divise l'objet en cellules et chaque échantillon contient plus qu'un index de cellule, mais seulement jusqu'à un nombre maximum *fixe*. Ceci nous permet d'utiliser une structure de données régulière (un tableau) moins coûteuse et complexe qu'une structure dynamique (un arbre de pointeurs). On alloue le tableau comme si le maximum de cellules étaient référencées pour chaque échantillon.

Il est à noter que nous avons choisi de subdiviser chaque objet séparément (une grille par objet), plutôt que de subdiviser toute la scène d'un seul coup. Il serait tout aussi valable de subdiviser toute la scène dans un bloc (voir section 3.5.2).

3.2.2 Préfiltrage

Dans le contexte du rendu par champ de lumière, un préfiltrage est nécessaire pour obtenir une bonne approximation de la valeur de radiance recherchée et pour que le flou introduit corresponde à la limite de la fréquence d'échantillonnage. Ceci est aussi vrai pour le champ de

visibilité, et il correspond à ajouter à un échantillon les index de tous les éléments de géométrie (i.e. les cellules) qui sont visibles dans la pyramide qui passe à travers sa paire (u, v) et (s, t) (voir figure 3.2(c)).

Le champ de visibilité est créé en lançant des rayons dans la scène à travers toutes les valeurs possibles de (u, v, s, t) sur les plans de paramétrisation. Au moment de créer le champ de visibilité, le nombre maximum d'index de cellule $N_{\text{max cells}}$ est fixé. Pour un échantillon, on suréchantillonne la scène en utilisant une grille de 3×3 positions sur les plans uv et st résultant en 81 rayons différents. On aurait pu aussi utiliser un algorithme de suréchantillonnage adaptatif [Whi80] [Coo86]. Comme un nombre limité de cellules est disponible pour le stockage de chaque échantillon, on trie les cellules que les rayons ont intersectées par le nombre de fois qu'elles ont été intersectées et on stocke ensuite les $N_{\text{max cells}}$ cellules qui ont été le plus souvent touchées. S'il y a moins que $N_{\text{max cells}}$ cellules visibles, on marque le reste des index comme "vide" (voir section 3.4).

On peut considérer cette technique comme un tampon d'objets [WHG84] (*item buffer*) généralisé à quatre dimensions pour pouvoir traiter plusieurs points de vue, et avec un nombre limité d'objets.

3.3 Algorithme de rendu

On effectue le rendu d'une nouvelle vue de la scène en ré-échantillonnant avec des rayons le champ de visibilité, depuis le point de vue de la caméra. Chaque échantillon touché nous donne une ou plusieurs cellules, et on marque ces dernières comme "visibles" dans une structure de grille similaire à celle utilisée pour stocker les polygones des objets. Ensuite, on parcourt cette structure de marquage, et on effectue seulement le rendu des polygones contenus dans les cellules marquées. Pour ceux-ci, on utilise le rendu "matériel" traditionnel (avec tampon de profondeur matériel SGI/OpenGL). Notre algorithme est décrit en pseudo-code à la figure 3.3.

En comparaison avec la technique de rendu habituelle, c'est-à-dire d'envoyer la totalité des polygones au pipeline graphique, notre technique de rendu ajoute un coût additionnel constitué de la discrétisation en pixels de l'intersection de la projection des plans de la paramétrisation sur le plan image, et du ré-échantillonnage (incluant toute forme de reconstruction, dans notre

```
begin
    Projeter les plans (u, v) et (s, t) sur le plan de vue
    Discrétiser en pixels l'intersection de ces plans sur le plan de vue
    foreach pixel dans l'intersection do
        Ré-échantillonner le champ de visibilité (pour trouver les cellules visibles)
        Marquer les cellules associées avec chaque échantillon
    end
    foreach cellule dans la grille de marquage do
        if la cellule est marquée then
            Effectuer le rendu de la géométrie de la cellule (avec OpenGL)
        end
    end
end
```

FIG. 3.3: Rendu avec champ de visibilité.

cas, l'interpolation bilinéaire ou quadrilinéaire qui correspond simplement à l'ajout potentiel de cellules à rendre).

Il faut mentionner qu'un polygone présent dans plusieurs cellules marquées sera envoyé au pipeline graphique plusieurs fois. En général, on choisit des résolutions de grilles de façon à ce que chaque cellule contienne beaucoup de polygones et donc peu de polygones occupent plus d'une cellule. Nous considérerons ce facteur comme négligeable dans notre analyse de coût.

La discrétisation en pixels pour effectuer le ré-échantillonnage n'est pas nécessairement effectuée dans la résolution de l'image. Ce choix de résolution n'influence pas la résolution de l'image finale produite, mais plutôt la qualité de l'estimation de visibilité obtenue, car c'est la résolution à laquelle on ré-échantillonne pour obtenir la liste de cellules visibles. On utilisera typiquement une résolution plus faible que celle de l'image à produire[5].

On obtient une réduction du temps de rendu si

$$K_{\text{discrétisation en pixels + ré-échantillonnage}} + t_{\text{rendu cellules visibles}} < t_{\text{rendu toutes cellules}}$$

où $K_{\text{discrétisation en pixels + ré-échantillonnage}}$ est un temps constant pour une certaine taille d'image

[5]Il est à noter que de choisir une résolution de ré-échantillonnage égale à celle de l'image à produire ne garantit pas une estimation de visibilité parfaite puisqu'au départ, la visibilité dans un échantillon n'est elle-même pas complète (à cause du nombre limité de cellules par échantillon) et parce que si on n'utilise pas l'interpolation en uv et st, il reste quand même un risque de ne pas marquer une cellule visible.

donnée[6]. Une comparaison avec le tracer de visibilité sera faite plus loin (section 3.3.2).

L'ampleur de ce gain en vitesse dépendra des facteurs suivants :

- **La quantité d'occlusion totale entre les cellules :** s'il y a plusieurs cellules complètement cachées, on évitera d'en effectuer le rendu. Plus il y a d'occlusion dans la scène en général, plus le gain en vitesse sera important, à cause de ce facteur ;
- **La nombre total de primitives :** le coût de la discrétisation en pixels et du ré-échantillonnage est constant, et est donc amorti sur le coût du rendu au total. S'il faut faire le rendu de beaucoup de polygones, le coût du ré-échantillonnage est négligeable par rapport au gain en vitesse par rapport à la géométrie dont on évite de faire le rendu ;
- **La résolution des grilles :** si on choisit une résolution très fine pour les grilles de cellules des objets, il sera plus coûteux de les parcourir toutes pour vérifier si elles sont marquées pour être rendues. Il serait peut-être possible d'utiliser une table de hachage lors du ré-échantillonnage pour stocker les index de cellules marquées. Ceci nous permettrait de faire une recherche rapide dans le but de ne pas stocker plusieurs fois un index de cellule, technique plus coûteuse que d'utiliser une simple grille de marquage, mais qui a comme avantage qu'on n'aurait pas à parcourir de grille de marquage pour faire le rendu.

La quantité d'occlusion partielle entre les cellules influence aussi le temps de rendu, dans le sens où une cellule partiellement visible sera envoyée au pipeline graphique au complet. On se trouve donc à calculer une solution de visibilité *conservatrice*, c'est-à-dire qu'on envoie parfois des surfaces cachées au pipeline graphique. La visibilité est ensuite traitée "correctement" au pipeline par un tampon de profondeur. Aussi, l'occlusion intra-cellule, c'est-à-dire l'occlusion à l'intérieur d'une même cellule n'est pas traitée spécialement. On envoie donc les surfaces d'une cellule qui sont cachées par d'autres surfaces dans la même cellule au pipeline graphique.

[6]Le temps dépend plutôt réellement de la taille de l'intersection des plans de la paramétrisation projetés sur le plan image, car on n'a pas besoin de discrétiser et ré-échantillonner les régions en dehors de cette intersection. On effectue cette optimisation en pratique.

3.3.1 Postfiltrage

Pour le rendu par champ de lumière, le postfiltrage est appliqué lorsqu'on ré-échantillonne les plans de la paramétrisation. Les 4 ou 16 échantillons adjacents sont utilisés pour interpoler une valeur pour le rayon.

De la même façon, on peut utiliser les échantillons de visibilité adjacents pour améliorer notre estimation de la visibilité selon un rayon. On considère les 4 (sur uv ou st) ou 16 (sur uv et st) échantillons adjacents aux points d'intersection du rayon avec les plans en faisant simplement l'union des cellules à marquer de ces échantillons. On marque donc plus de cellules que ne seraient marquées par un ré-échantillonnage ponctuel[7]. Par contre, on court le risque d'envoyer plus de géométrie cachée au pipeline graphique. Le postfiltrage implémenté de cette façon permet de réduire le nombre de trous tel que discuté à la section 3.5.2.

3.3.2 Tracer de visibilité

Pour comparer correctement les performances de notre algorithme, nous avons implémenté une version de l'algorithme de rendu qui trace des rayons pour déterminer les cellules visibles *au moment du rendu*. La différence entre cet algorithme et l'algorithme présenté à la figure 3.3 consiste alors en l'échantillonnage. Cet algorithme nous permet d'évaluer l'ampleur du gain en vitesse dû au précalcul. Il est par contre difficile de comparer les qualités des estimations de visibilité produites (voir section 3.5 pour plus de détails).

3.3.3 Exploitation de la cohérence temporelle

Pour tenter de réduire encore plus le nombre de trous, dans le cas d'une animation de la caméra (mouvement interactif), une technique simple consiste à conserver marquées les cellules qui ont été marquées pour les n rendus précédents. Nous avons aussi implémenté cette technique[8]. Cette méthode fonctionne très bien, et son coût en temps d'exécution est négligeable : il suffit d'allouer une liste d'autant de grilles de marquage que de rendus pour lesquels on veut conserver les cellules marquées. Pour chaque cellule on marque toutes les

[7] C'est-à-dire où on utilise le plus proche échantillon voisin.
[8] Les résultats seront par contre calculés sans cette méthode.

grilles dans la liste et on utilise les marques de la dernière grille comme estimation de la visibilité. Ensuite on réinitialise cette dernière grille et on la place au début de la liste.

3.4 Implémentation

L'implémentation de ce logiciel a été effectuée en C++, et les temps de calcul donnés ont été mesurés sur un SGI avec un processeur MIPS R10000 et 128 MB de mémoire vive. Nous avons construit une interface-usager à l'aide de la bibliothèque *Qt* [TT98].

La compression pour le champ de visibilité n'a pas été implémentée. Même si la méthode pourrait être généralisée pour n'importe quel type de primitive, notre implémentation ne supporte présentement que des polygones (triangles et quadrilatères).

Chaque échantillon contient un tableau d'index de cellule. Un index de cellule est constitué d'un index à l'objet qui contient la cellule (ou -1 si la référence est "vide"), ainsi que les coordonnées de la cellule dans la grille associée à cet objet (voir figure 3.4).

```
typedef struct {
    unsigned char meshidx ; /* index de l'objet */
    unsigned char cx ; /* coordonnées des cellules */
    unsigned char cy ;
    unsigned char cz ;
} EchantillonVisibilite;
```

FIG. 3.4: Structure d'un échantillon de visibilité.

Le système fonctionne à des vitesses interactives et offre la possibilité d'effectuer le rendu sur plusieurs objets, ainsi que plusieurs champs de visibilité (plusieurs paramétrisations).

3.5 Résultats

Nous avons testé notre algorithme sur quelques scènes. Un rendu par tampon de profondeur ordinaire d'une scène constituée de six tortues chinoises[9] (la scène *turtle*) est illustré à la figure

[9]Le modèle de tortue provient du projet de reconstruction du jardin Yuan Ming Yuan (*Garden of Perfect Brightness*) à l'université de Colombie-Britannique [XX98].

3.5. Chaque modèle de tortue comprend 38510 triangles, et il y a 6 instances de tortues (la scène contient donc en tout 231060 triangles). On divise chaque tortue séparément par une grille de $12 \times 12 \times 12$ ce qui fait en tout 10368 cellules pour l'ensemble des tortues. Après avoir séparé les polygones dans les grilles de cellules, il ne reste que 4680 cellules qui contiennent des polygones (les autres étant vides).

Le plan uv de la paramétrisation est très grand et est positionné à l'infini (derrière le point de vue) et le plan st est à peu près de la même taille que l'objet et tout près de ce dernier (représenté par le cadre dans la figure). On a créé le champ avec une résolution de 16×16 pour les deux plans.

Une image produite par notre algorithme de rendu (c'est-à-dire en effectuant seulement le rendu des cellules marquées), sans postfiltrage (on marque les cellules associées à l'échantillon correspondant au plus proche uv du point d'intersection du rayon avec le plan uv et au plus proche st du point d'intersection du rayon avec le plan st), et avec un seul index de cellule par échantillon est illustrée à la figure 3.6. Le ré-échantillonnage est effectué à une résolution de 1 pixel par bloc de 4 pixels (la moitié de la résolution de l'image).

On voit beaucoup de trous dans cette figure : ils correspondent à des cellules de polygones non-rendues. La basse résolution choisie ici pour la paramétrisation sert à bien mettre en évidence ces artefacts. À la figure 3.7, une image de la même scène mais qui utilise cette fois le postfiltrage sur uv et st décrit à la section 3.3.1 est illustrée. On pourrait aussi utiliser un postfiltrage seulement sur le plan uv, ou seulement sur le plan st, et on obtient un effet similaire (i.e. plus de cellules visibles que sans postfiltrage), mais moins important. En comparant les figures 3.6 et 3.7, on remarque que même avec le postfiltrage, il reste encore des trous. Le fait d'utiliser plus qu'un index de cellule par échantillon améliore grandement l'estimation de visibilité. Ceci est illustré à la figure 3.8, où on utilise cinq index de cellule par échantillon, et aucun postfiltrage.

Il est important de noter que les techniques de postfiltrage et d'utilisation de plusieurs cellules par échantillon se combinent ensemble naturellement pour améliorer encore plus l'estimation de visibilité obtenue.

À la figure 3.9, la scène est rendue depuis un point de vue de côté, avec seulement la

FIG. 3.5: Rendu traditionnel (toute la géométrie).

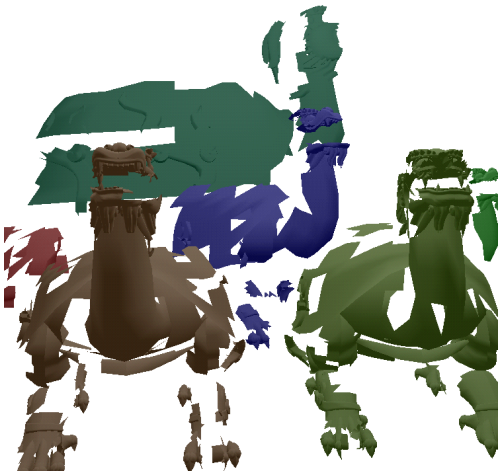


FIG. 3.6: Rendu avec notre algorithme : Sans postfiltrage, une cellule par échantillon.

FIG. 3.7: Rendu avec notre algorithme : Postfiltrage uv et st.

FIG. 3.8: Rendu avec notre algorithme : Sans postfiltrage et avec cinq cellules par échantillon.

FIG. 3.9: Vue de côté des cellules traitées depuis un point de vue juste en face de la scène de six tortues chinoises.

géométrie qui serait marquée et rendue depuis un point de vue situé juste devant l'objet. Une comparaison similaire est illustrée aux figures 3.10 et 3.11 à la page 41 pour la scène *cars*. On peut observer que la plupart des cellules cachées sont correctement ignorées par notre algorithme.

On retrouve des statistiques aux figures 3.13 et 3.14 à la page 48 ainsi qu'aux tableaux 3.1 et 3.2. Il est à noter que comme la scène *bunny* (figure 3.12) ne consiste que d'un objet, et n'a que peu d'occlusion entre ses cellules, seulement environ la moitié de la géométrie est rendue (la partie visible). Une proportion beaucoup plus petite de cellules est rendue pour des scènes avec modérément d'occlusion.

Nous comparons notre méthode avec deux techniques différentes pour le calcul de visibilité :

- *Rendu par tampon de profondeur ordinaire.* Envoyer toute la géométrie au pipeline graphique, sans utiliser de listes d'affichage (de l'anglais *display lists* [NDW93]). Ces dernières fausseraient les comparaisons (les listes d'affichage offrent une accélération importante[10]).
- *Tracer de visibilité.* Lancer interactivement des rayons vers les objets pour déterminer la visibilité juste avant de faire le rendu des polygones (voir section 3.3.2). Pour obtenir une accélération, le gain en géométrie évitée doit contre-balancer le coût de faire ce lancer de rayons.

Comparer notre algorithme avec le tracer de visibilité nous permet d'estimer la performance du précalcul. L'algorithme de tracer de visibilité nous donnera des résultats exacts[11] si on utilise une résolution égale à la résolution de l'image à produire. En fait, à cette résolution, le tracer de visibilité sera plus lent qu'un simple lancer de rayons (*ray casting*) avec calculs d'illumination à cause du rendu par tampon de profondeur subséquent, mais il faut garder à l'esprit que si la scène contient un très grand nombre de polygones, même un tracer de rayons pourra être plus rapide qu'un rendu traditionnel par balayage de lignes avec tampon de profondeur.

En général on devra utiliser le tracer de visibilité à un niveau de sous-échantillonnage assez bas pour maintenir des temps de rendu comparables avec notre algorithme. Sous-échantillonner dans ce contexte est équivalent à faire le rendu à une résolution donnée mais d'échantillonner pour la visibilité à une résolution plus basse (beaucoup plus basse, en fait, car on utilisera typiquement un ratio de 1:10, donc un rayon par 100 pixels pour déterminer la visibilité). On obtiendra donc aussi des trous pour le tracer de visibilité.

Par contre, en considérant le coût considérable de faire le tracer de visibilité, un rendu avec notre algorithme de champ de visibilité et une bonne paramétrisation donnera de meilleurs résultats que cet algorithme, pour une vitesse équivalente.

Il est difficile de comparer directement la qualité de l'estimation de visibilité produite par ces deux algorithmes (i.e. le nombre et la taille des trous). Dans le cas du champ de visibilité, la

[10] Cependant, il est important de garder à l'esprit qu'une implémentation de notre technique ne pourrait pas utiliser de listes d'affichage en matériel, à cause du grand nombre de listes qu'on aurait à créer (une par cellule).

[11] Au moins aussi exacts que ceux d'un tampon de profondeur (le tampon de profondeur est aussi un échantillonnage).

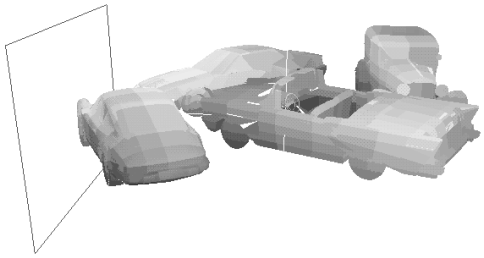


FIG. 3.10: Rendu de la scène *cars* (toute la géométrie).

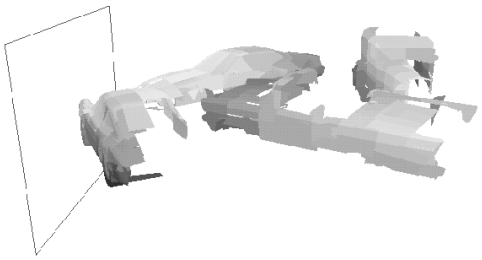


FIG. 3.11: Rendu de la scène *cars* (cellules qui seraient visibles selon un point de vue situé derrière le plan à gauche).

qualité dépendra de la résolution de la paramétrisation, de la résolution du ré-échantillonnage, ainsi que de la taille et du positionnement des cellules dans la scène. Dans le cas du tracer de visibilité, la qualité dépendra de la résolution d'échantillonnage.

Les résultats ont été analysés en regard des critères suivants :

- *La quantité de géométrie rendue (le nombre de cellules).* Voir la figure 3.13 à la page 48 et les tableaux 3.1 et 3.2. Pour la scène *turtle*, ces statistiques montrent que typiquement, moins de 800 cellules qui contiennent de la géométrie[12] sont rendues sur un total de 4680 cellules non-vides (moins que 20%). Au tableau 3.2 le nombre de cellules rendues avec l'interpolation uv est légèrement plus grand que le nombre de cellules rendues avec l'interpolation $uv + st$. Ceci est dû à la variance dans les résultats (les résultats offerts sont une moyenne pour une série d'images à partir de plusieurs points de vue de la scène).
- *Temps de rendu (*sans *listes d'affichage)*. Notre algorithme peut difficilement prendre avantage des listes d'affichage, car la liste de cellules rendues varie selon le point de vue. Aussi, il n'y a typiquement pas un nombre suffisant de listes d'affichage supporté par le matériel pour en utiliser une par cellule. Les temps de rendu pour les scènes *turtle* et *cars* sont reportés à la figure 3.14 et au tableau 3.1, respectivement. Pour la scène *turtle*, on observe une accélération quand on sous-échantillonne à 1 :3 résolution, même avec le postfiltrage en $uv + st$ (à comparer avec le cas où on échantillonne à la résolution de l'image désirée, ce qui n'apporte pas d'accélération).

[12] Pour calculer le total, on ne compte évidemment pas les cellules vides.

Les listes d'affichage offrent généralement une bonne accélération pour le rendu ordinaire, mais le temps de rendu croît quand même de façon linéaire avec le nombre de primitives. On peut donc espérer que même sans listes d'affichage, il y aura un nombre de primitives où notre technique sera plus rapide que le rendu ordinaire. De plus, des scènes typiques exhibent beaucoup plus d'occlusion que dans nos scènes de test, ce qui ne peut qu'améliorer le temps relatif de rendu de notre algorithme par rapport au rendu ordinaire. Par exemple, dans le cas de scènes de ville ou encore à l'intérieur d'édifices, une très faible proportion de la scène est visible à tout moment.

Interpolation	Cellules	% cellules	Temps	% temps
(aucune)	231.4	17.0%	142 ms/fr.	31.1%
uv	283.8	20.8%	181 ms/fr.	39.6%
st	251.6	18.4%	194 ms/fr.	42.4%
$uv + st$	341.4	25.0%	385 ms/fr.	84.2%

TAB. 3.1: Statistiques de rendu pour la scène *cars*, $uv = 12 \times 12$, $st = 48 \times 48$, 60064 triangles, 1364 cellules, un index de cellule par échantillon, temps de rendu traditionnel : 457 ms/*frame*.

Interpolation	Cellules	% cellules
(aucune)	91.2	38.8%
uv	104.9	44.6%
st	91.3	38.9%
$uv + st$	104.5	44.5%

TAB. 3.2: Statistiques de rendu pour la scène *bunny* (peu d'occlusion), $uv = 8 \times 8$, $st = 32 \times 32$, 3444672 triangles, 235 cellules, un index de cellule par échantillon.

Les résultats furent calculés en faisant une animation de la caméra (la scène est statique) et en faisant la moyenne des mesures. Bien que la technique permette d'utiliser plusieurs paramétrisations, nous nous sommes limités à une seule paire de plans pour nos tests. Nous avons donc limité la caméra aux positions/directions où on peut voir la scène en entier, c'est-à-dire où les projections des plans uv et st s'intersectent complètement (si on tente d'échantillonner en dehors de la paramétrisation, on ne marque pas de cellules et ceci fausserait les résultats).

3.5.1 Résolution

Il est important de noter que nous avons utilisé des résolutions assez basses comparées à celles requises pour faire un rendu décent dans le cadre du rendu par champ de lumière. Les résolutions choisies dépendent de la résolution des grilles de cellules, du positionnement des plans de paramétrisation, du positionnement relatif des objets par rapport à ces derniers, ainsi que de la résolution du ré-échantillonnage. Pour avoir une idée de la résolution relative utilisée en champ de visibilité, nous avons effectué un rendu par champ de lumière à des résolutions comparables à celles qu'on utilise avec notre technique (voir figure 3.12). Cette figure montre un rendu de la scène *bunny* avec des résolutions de 8×8 en uv, 32×32 en st, et sans postfiltrage, pour mettre en évidence les discontinuités dues à la basse résolution.

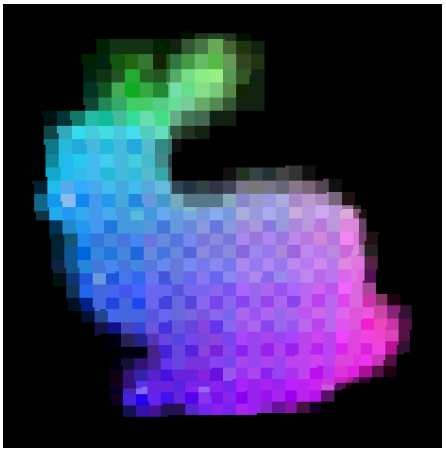


FIG. 3.12: Rendu de la scène *bunny* par champ de lumière à résolution équivalente (aucun postfiltrage).

3.5.2 Discussion (effets des paramètres)

Nous décrivons dans cette section les problèmes reliés à notre technique de rendu.

Taille mémoire

Comme pour le rendu par champ de lumière, la taille du champ de visibilité occupe beaucoup de mémoire et ce, même si nos résolutions sont un peu plus raisonnables. Cette technique

n'est donc pas indiquée pour calculer plusieurs images d'une animation où des objets sont en mouvement : la technique, telle que nous l'avons présentée, est donc limitée aux scènes statiques où seule la caméra bouge.

En ce qui concerne la compression en mémoire, on estime qu'il y a beaucoup plus de cohérence pour le champ de visibilité que pour le champ de lumière. Deux index de cellules adjacentes auront souvent la même valeur numérique (les index). On estime que les techniques de compression donneront de meilleurs résultats sur ce genre de données.

Occlusion partielle et trous

Comme la fonction de visibilité contient beaucoup de discontinuités, il est difficile de l'échantillonner correctement. Par exemple, s'il y a de véritables petits trous dans un objet à travers lesquels un autre objet est visible, si les trous sont suffisamment petits et qu'aucun rayon d'échantillonnage ne passe par le trou lors de la création du champ de visibilité, nous n'effectuerons jamais le rendu des surfaces placées derrière le trou. Aussi, à cause du nombre limité d'index de cellule par échantillon, même si un rayon atteignait la surface derrière le trou lors de la création du champ, la cellule de cette surface ne serait peut-être pas considérée assez importante par notre procédure de triage pour être enregistrée dans l'échantillon (voir section 3.2.2).

Similairement au problème de l'occlusion partielle, des cellules qui ne contiennent que peu ou encore de petits polygones pourraient être ratées complètement par l'échantillonnage lors de la création du champ (elle ne seront alors simplement jamais rendues et ce, même avec le meilleur ré-échantillonnage possible).

Ces deux problèmes, combinés aux problèmes du ré-échantillonnage, causent les trous visibles dans les images qui sont dus à la géométrie non-rendue dans les modèles. Le suréchantillonnage et l'utilisation de multiples index de cellule aide à réduire l'ampleur de cet effet, mais pas complètement.

De plus, lorsqu'on effectue un rendu depuis des points de vue situés loin de la paramétrisation, moins de rayons de la caméra l'intersectent et seulement un plus petit sous-ensemble des cellules visibles sont rendues. Ce problème est contre-balancé par le fait que les

images produites sont plus petites.

Positionnement des plans et effets de la paramétrisation

Le succès de la technique dépend aussi du positionnement des plans de paramétrisation dans l'espace. Différentes façons d'orienter les plans vont couvrir différents sous-ensembles de lignes, plus ou moins efficaces selon les points de vue recherchés.

Visibilité approximative

Notre technique ne peut donner qu'une solution approximative de la visibilité. Bien que la majorité des méthodes de calcul de visibilité cherchent à trouver une solution exacte, certaines techniques permettent aussi de calculer une visibilité approximative (par exemple avec les *HOMs* de Zhang *et al.* [ZMHI97]).

3.6 Travaux futurs

La méthode n'est pas parfaite. D'abord et avant tout, le problème des trous dus à l'échantillonnage se présente comme l'aléa principal de notre technique. Telle que développée jusqu'ici, notre méthode ne peut pas calculer la visibilité de façon exacte, c'est-à-dire que l'ensemble de polygones qu'on calcule comme visibles peut contenir des polygones invisibles (acceptable), et des polygones visibles peuvent manquer (inacceptable). Selon la géométrie de la scène et selon l'application, les trous peuvent être considérés plus ou moins importants. Même si on choisit des résolutions un peu plus grandes que nécessaires, on obtient de toutes façons une solution conservatrice de visibilité (voir section 3.1). Il est important de noter que nos résultats ont été calculés avec des résolutions basses de façon à bien démontrer les problèmes reliés avec notre technique, ainsi qu'à mettre en évidence les améliorations apportées.

Une amélioration consisterait à préfiltrer hiérarchiquement le champ de visibilité, comme un *mip-map* [Wil83], mais en quatre dimensions. Ceci pourrait résoudre le problème de sous-échantillonnage du champ quand la caméra est loin de la paramétrisation (voir section 3.5.2).

Comme les échantillons des niveaux plus élevés que le niveau de base du *mip-map* devraient contenir plus d'index de cellule par échantillon pour correctement représenter la géométrie visible, on décuplerait directement la taille du champ par le nombre de niveaux de *mip-map* supportés et rendrait ainsi l'utilisation du *mip-map* prohibitive. Pour s'en sortir, on pourrait utiliser des grilles hiérarchiques et on stockerait les index de plus hauts niveaux de hiérarchie dans les échantillons des niveaux plus hauts du *mip-map*, de façon à ce que la taille d'un échantillon reste constante indépendemment du niveau de *mip-map*. Comme le champ de visibilité est une structure en quatre dimensions, un *mip-mapping* ne coûterait que

$$\sum_{i=0}^{\infty} \frac{1}{(2^4)^i} = \frac{1}{1-\frac{1}{16}} = \frac{16}{15}$$

de la taille du champ, ce qui est négligeable par rapport à l'amélioration apportée.

Aussi, on pourrait tenter d'améliorer la structure des grilles elles-mêmes, en utilisant des *octrees*, avec des cellules dont la taille dépend de la distance aux paramétrisations. Ceci permettrait de compenser pour les cellules qui ont moins de chances d'être échantillonnées en augmentant leur taille. Par contre, l'accès aux noeuds d'un *octree* est plus coûteux que pour une grille (si on veut stocker l'*octree* en espace optimal). De plus, si on restreint les points de vue à une zone de l'espace, il serait intéressant d'explorer l'utilisation de grilles déformées ou orientées différemment.

Il serait également intéressant d'explorer l'utilisation de multiples valeurs le long d'un rayon, un peu comme le font Shade *et al.* [SGHS98]. En stockant la profondeur avec chaque échantillon on pourrait peut-être lever la contrainte de pouvoir visualiser vers une région convexe ou encore depuis une région convexe.

La compression du champ de visibilité n'a pas été étudiée. Plusieurs index de cellules adjacentes sont probablement souvent identiques, à cause de la cohérence du modèle et de la paramétrisation. On peut donc espérer un haut taux de compression.

Finalement, plus de travail pourrait être effectué pour palper la qualité des résultats produits. Il est difficile de juger de l'efficacité de l'algorithme en terme d'exactitude en calculant le pourcentage de cellules non-rendues, car ce rapport dépend de la quantité d'occlusion à partir d'un point de vue précis. Il serait plus intéressant de comparer les artefacts en espace image

(peut-être en comparant les pixels avec une image de référence), pour voir de combien diffère notre solution par rapport à la solution exacte.

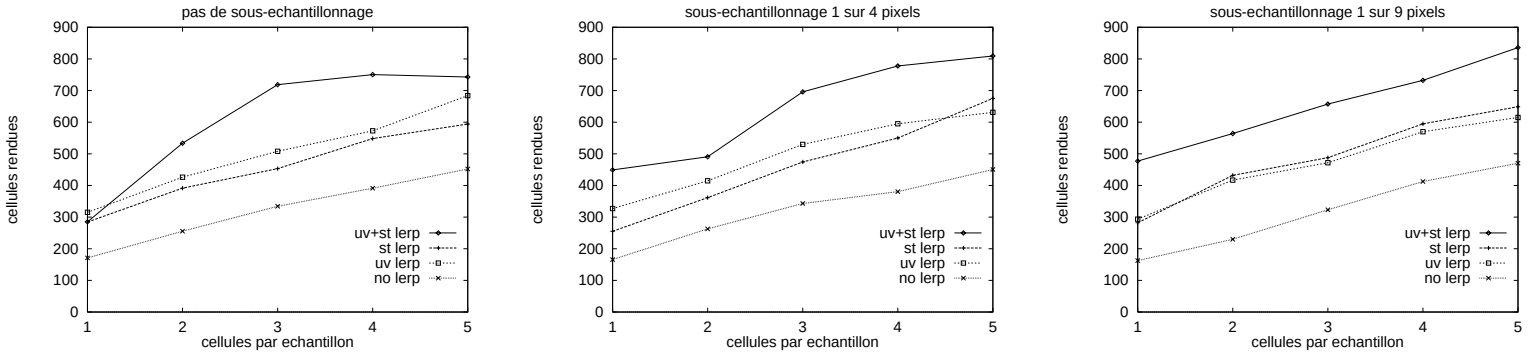


FIG. 3.13: Nombre de cellules rendues pour la scène *turtle*, $uv = 16 \times 16$, $st = 16 \times 16$, 231060 triangles, 4680 cellules.

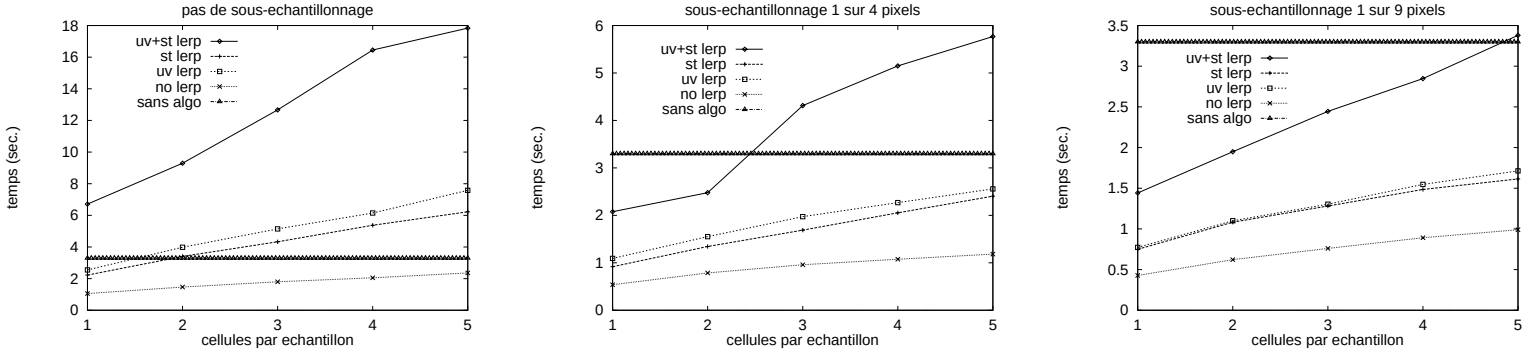


FIG. 3.14: Temps de rendu pour la scène *turtle*, $uv = 16 \times 16$, $st = 16 \times 16$, 231060 triangles, 4680 cellules, temps de rendu traditionnel : 3.3 secs.

Chapitre 4

Ré-illumination de champ de lumière

Un problème important qui se pose dans le contexte du rendu par champ de lumière repose dans l'insertion d'objets de ce type dans des scènes, plus particulièrement au niveau de l'interaction de l'illumination présente dans la scène avec ces objets.

A priori, un champ de lumière contient des valeurs de radiance qui reflètent l'illumination telle que mesurée au moment de la capture de l'objet. Ceci est un désavantage sérieux de la technique car l'illumination y est alors fixe. Le problème que nous nous proposons d'explorer dans cette excursion est de modifier l'illumination d'un objet rendu par champ de lumière et de refléter ces changements dans l'apparence de l'objet.

Nous étudierons plus en détail les conditions de cette ré-illumination, et modifierons la technique de Levoy et Hanrahan [LH96] pour permettre un certain degré de liberté dans le changement de l'illumination de la scène.

Le domaine de la vision par ordinateur a traité le sujet de l'élimination des reflets spéculaires sur des images. Nous ferons un rappel de ces notions à la section 4.1. Cette section contient aussi un survol plus général du domaine de l'intégration d'images réelles avec des images synthétiques, la réalité augmentée par ordinateur (*CAR*[1]), car ce champ est aussi concerné par la modification de l'illumination dans des images réelles. Nous allons ensuite définir clairement le problème à résoudre (section 4.2.1), ainsi qu'établir un contexte pour la ré-illumination de champs de lumière (modèle d'illumination, calculs de propriétés de l'objet dont on a besoin,

[1] De l'anglais : *Computer Augmented Reality*.

etc.). Les résultats de nos expérimentations seront présentés en section 4.5, et nous conclurons avec une discussion des problèmes reliés à notre technique ainsi qu'une discussion des multiples possibilités qui s'ouvrent à nous pour faire des travaux futurs dans ce domaine.

4.1 Travaux antérieurs en ré-illumination

Quelques travaux ont déjà été présentés dans le but de modifier l'illumination pour le rendu basé sur des images. Par contre, ce champ est tout récent, et quelques domaines connexes fournissent d'intéressantes contributions pour notre travail. Plus particulièrement, le domaine de la réalité augmentée par ordinateur fournit plusieurs indices quant aux problèmes à résoudre. Aussi, nous résumerons les travaux récents en rendu basé sur des images à la section 4.1.1, les travaux reliés à la réalité augmentée par ordinateur à la section 4.1.3, et finalement, quelques travaux pertinents du domaine de la vision (section 4.1.4).

4.1.1 Travaux antérieurs en ré-illumination de champ de lumière

Deux méthodes ont été proposées pour changer l'illumination de rendus par images. D'abord, Wong *et al.* [WHON97] présentent une technique originale pour le rendu par champ de lumière avec 2PP : considérer les pixels sur le plan uv comme ayant une BRDF "apparente". Comme le plan uv n'est pas généralement incident avec une surface réelle, on ne peut pas correctement parler de BRDF, mais on considère de façon similaire l'illumination qui quitte dans toutes les directions ce point dans l'espace, ce qui forme une structure similaire à une BRDF. Au lieu de stocker une seule valeur de radiance pour chaque échantillon correspondant à un rayon, on stocke plusieurs valeurs, une pour chaque direction possible de la lumière incidente. Chaque pixel sur le plan uv a une valeur qui dépend de la valeur de s, t et de θ, ϕ, la direction incidente de la lumière. Le champ résultant comporte six dimensions : u, v, s, t les intersections sur les plans de la paramétrisation et θ, ϕ la direction incidente de la lumière. Comme cette nouvelle structure de données requiert beaucoup d'espace mémoire, ils encodent la BRDF "apparente[2]" sous forme d'harmoniques sphériques [SAWG91]. Ils utilisent ensuite la propriété de superpo-

[2]Unidirectionnelle dans ce contexte, pour un s, t fixe.

sition de la lumière pour varier la position de lumières directionnelles. Comme cette méthode ne stocke pas la position des échantillons dans l'espace, elle ne peut pas traiter les lumières ponctuelles correctement[3]. De plus, la création de ce type de champ de lumière étendu avec des objets réels s'avérerait assez difficile : il faudrait avoir le contrôle sur la position et la direction de l'illumination pour approximer des lumières directionnelles et sur le type de lumière utilisée, et faire varier ce paramètre (θ et ϕ pour la lumière) en plus de faire varier la position et la direction de la caméra. La méthode fonctionne en des temps interactifs (ils utilisent l'apposition de textures par le matériel comme Gortler *et al.* [GGSC96]), et peuvent traiter les ombres de l'objet sur lui-même (car elles sont considérées dans le précalcul).

Une autre méthode, introduite par Oliveira et Bishop [OB98], consiste à étendre la modélisation plénoptique [MB95] pour y ajouter les effets suivants : l'illumination ajoutée, les ombres, et les effets spéculaires de premier ordre. Cette technique assume qu'on a à notre disposition la disparité (l'inverse de la profondeur) pour chaque pixel. On doit d'abord effectuer une reconstruction partielle[4] de la géométrie, en connectant les pixels adjacents si leur disparité ne varie pas trop. On utilise ensuite cette géométrie partielle pour vérifier la visibilité avec les lumières ajoutées, pour projeter des ombres. Lorsqu'on ré-illumine un pixel de l'image désirée, on vérifie l'occlusion pour chaque lumière (stockée en précalcul pour chaque lumière (fixe) et pixel), et on ajoute l'illumination de celle-ci. Une esquisse de leur algorithme est présentée à la figure 4.1. Autrement, la technique est similaire à celle décrite par McMillan et Bishop [MB95]. Il est aussi suggéré qu'on pourrait faire le rendu de la géométrie à partir de la position d'une image de référence et *ensuite* de déformer cette image pour obtenir l'image désirée (comme pour l'interpolation de vues).

Il y a plusieurs problèmes potentiels avec cette méthode. Premièrement, tous les problèmes de trous associés à l'interpolation d'images sont présents. Deuxièmement, il est difficile d'extraire la géométrie de façon précise à partir d'images de profondeur grossières [TL94].

Il est important de noter qu'aucune de ces techniques ne nous permet de modifier l'illumination déjà présente dans le champ de lumière. Ceci représente un problème de vision difficile

[3] La position de la surface sur le rayon de l'échantillon considéré est nécessaire pour calculer le vecteur L, de l'intersection à la lumière.

[4] On obtient en sortie les surfaces visibles connectées (en anglais : *connected components*).

```
begin
    foreach image de référence do
        Générer son maillage connecté
        Calculer la position 3D des pixels
        Calculer la normale aux pixels
        Simplification du maillage (optionnel)
    end
    foreach source de lumière do
        (B) begin
            Dessiner tous les maillages 3D au tampon image secondaire
            Marquer les pixels des maillages qui sont visibles depuis la source de lumière
        end
    end
    —Itérations—
    if la position d'une lumière a changé then
        Ré-initialiser les pixels visibles de cette lumière
        Exécuter (B)
    end
    foreach pixel de l'image désirée do
        foreach source de lumière do
            if le pixel est illuminé then
                Accumuler la contribution de la lumière au pixel
            end
        end
    end
end
```

FIG. 4.1: Algorithme d'illumination dynamique pour modélisation plénoptique [OB98].

à résoudre [SHW92].

4.1.2 Ré-illumination d'images

Haeberli [Hae92] présente une idée intéressante reliée au principe de superposition de l'illumination. Il capture plusieurs images d'une scène réelle, dans lesquelles il n'allume qu'une seule lumière à la fois, et recombine ensuite les images pour produire une illumination variable. Notamment, il considère la soustraction d'images (illumination négative).

Nimeroff *et al.* [NSD94] présentent une technique pour faire varier l'illumination d'une scène sans refaire le rendu. L'idée présentée est basée sur la propriété de superposition de l'illumination [BK96] : additionner ou multiplier l'intensité d'une source de lumière correspond

à additionner ou multiplier l'intensité de la contribution de cette lumière dans chaque pixel[5]. Ce travail considère l'illumination naturelle (e.g. du ciel), et exprime l'illumination sous forme de bases orientables (*Steerable filters* [FA91]) qui sont utilisées pour combiner linéairement des images de rendu précalculées dans le but de reproduire une image avec d'autres conditions d'illumination (sans refaire le rendu).

4.1.3 Réalité augmentée par ordinateur

Le champ de la réalité augmentée, qui traite de l'intégration d'images réelles et d'images synthétiques, a dû considérer la ré-illumination et l'addition des ombres dans les images réelles [FGR93]. Le but recherché est de "fondre" dans une même séquence des objets virtuels et une séquence filmée. La technique utilisée consiste à construire un modèle grossier de la scène réelle et de l'utiliser pour l'algorithme de rendu (dans ce cas-ci, la radiosité) avec les objets virtuels. Les propriétés de surface sont estimées à partir des intensités moyennes dans les images. Cette technique a été étendue par Drettakis *et al.* [DRB97], notamment, pour le rendu incrémental, ainsi qu'au niveau de la calibration des caméras pour aligner les objets reconstruits avec la scène réelle.

Selon Fournier *et al.* [Fou95], il y a cinq défis principaux à relever pour intégrer des objets virtuels dans une scène réelle : appliquer les ombres (dans les deux sens), l'estimation des illuminants dans la scène réelle [Gun91] (déterminer la position des sources de lumières), l'illumination globale commune, et le calcul des propriétés de surface à partir des images.

Pour appliquer les effets d'illumination des objets virtuels sur l'image réelle, on peut faire le rendu de la scène réelle approximative avec et sans les objets virtuels, et appliquer la différence sur l'image réelle (rendu différentiel [Deb98]).

4.1.4 Élimination des ombres et des reflets spéculaires dans une image

Comme un champ de lumière peut être considéré comme une série d'images, les méthodes de ré-illumination d'images ont un intérêt particulier pour notre application.

[5]Ce principe repose sur une relation linéaire entre l'illumination et les couleurs des pixels. Il est important de noter que bien que ce soit valide en réalité, cette supposition est erronée si on travaille à partir de photographies, à cause des différentes étapes de transfert non-linéaires impliquées dans le processus photo [DM97].

La ré-illumination d'images a été approchée dans le contexte de la composition d'images [PD84] [NINT89]. Quand on compose plusieurs images ensemble, on voudrait idéalement ajuster l'illumination de chacune des images pour donner l'impression d'une illumination commune cohérente dans la scène. De plus, les ombres doivent être ajoutées ou enlevées pour introduire les indices visuels entre les objets dans des images différentes. Pour la composition d'une seule image, cette tâche est généralement effectuée par un artiste, par un travail de retouches manuelles. Il serait par contre assez fastidieux d'effectuer cette tâche de façon cohérente pour une série d'images (par exemple, dans un clip vidéo, ou encore plus pour un champ de lumière).

La détection et l'élimination des ombres ont été étudiées plus en détail dans la littérature de vision par ordinateur, surtout en ce qui concerne les photographies aériennes (par exemple, pour les systèmes de guidage de missiles). On utilise typiquement une combinaison de détection de contours et de seuillage.

La détection et l'élimination des reflets spéculaires dans les images ont aussi été considérées dans la littérature. Lee *et al.* [Lee88] [LBS90] ont développé la méthode de *convergence de chromaticité* pour calculer la couleur d'un illuminant à partir de l'illumination d'objets, pour détecter les spécularités. D'autres ont utilisé une représentation des couleurs en espace RGB tridimensionnel pour faire une recherche des distributions en forme de "L" [Ger87] ou de "T" [KSK88] dans le but de séparer une image couleur en ses composantes diffuses et spéculaires. Ces méthodes suggèrent une technique naturelle pour éliminer la composante spéculaire : extrapoler dans l'espace RGB les couleurs des pixels correspondant aux reflets spéculaires en utilisant l'ensemble des couleurs diffuses détectées dans l'image.

Par contre, ces techniques ne fonctionnent pas très bien avec des images texturées, car la distribution des couleurs ne suit pas un motif prévisible. Romanzin [Rom95] résume les différents problèmes reliés à ces techniques.

4.2 Ré-illumination de champ de lumière

Dans cette section, nous décrirons les buts que nous nous sommes fixés, ainsi que notre approche au problème.

4.2.1 Problématique

Dans le cadre de ce projet, nous étudions le problème du changement d'illumination des objets représentés par champ de lumière. On ne considère pas les relations de visibilité entre les objets (objets par champ de lumière ou autres), ni les changements de propriétés de surface des objets par champ de lumière. Aussi, on ne considère que les objets statiques seulement (pas d'animation). On attaque le problème 2 présenté dans l'introduction (cf. section 1.2).

Voici une formulation du problème à résoudre :

> *Étant donné un champ de lumière quadridimensionnel (potentiellement augmenté avec plus d'information), modifier la technique de rendu pour permettre de changer l'illumination.*

4.2.2 Rendu avec illumination ajoutée

On peut définir deux variantes du problème :

1. Ré-illumination avec de nouvelles sources de lumière, en plus de l'illumination déjà capturée dans le champ de lumière (extraction ou spécification des propriétés de surface et ensuite ajout de l'illumination due aux nouvelles sources de lumière) ;
2. Ré-illumination avec un ensemble de sources de lumière complètement différent des sources présentes lors de la capture du champ donné (extraction ou spécification des propriétés de surface, extraction ou spécification des illuminants, dé-illumination et ensuite ré-illumination).

La deuxième technique suppose qu'on soit capable d'enlever l'illumination présente dans le champ de lumière. Il faudrait pour cela pouvoir résoudre le problème du rendu inverse[6], c'est-à-dire de retrouver les propriétés de surface des objets à partir d'images. Comme le champ de lumière nous fournit beaucoup plus d'échantillons qu'une image, il serait peut-être possible d'appliquer des techniques nous permettant de retrouver la BRDF des surfaces. Ceci sera discuté plus longuement à la section 4.7.

[6] Il faut faire attention à ne pas confondre le problème de rendu inverse [PF95], où on essaie de trouver les propriétés de surfaces à partir de couleurs, avec le problème d'illumination inverse où on essaie de spécifier les paramètres et positions des sources de lumière à partir de spécifications d'ombres ou de reflets spéculaires désirés en espace image (voir par exemple [PRJ97], nonobstant le coefficient de rugosité retrouvé dans cet article).

Nous restreindrons notre travail à l'ajout de nouvelles sources de lumière seulement, en spécifiant les propriétés de surface (c'est-à-dire que nous ne porterons pas particulièrement notre attention à retrouver les propriétés de surface originales de l'objet). De plus, on se limitera à des sources de lumière ponctuelles, bien que notre méthode puisse traiter des sources directionnelles (i.e. à l'infini) ou encore des projecteurs (lumières *spots*).

Pour des champs de lumière créés à partir d'objets synthéthiques, on peut avoir directement accès aux propriétés de surface des objets. On entrevoit alors deux méthodes de stockage pour le champ de lumière :

- Stocker la radiance issue de l'objet dans les échantillons sous une illumination fixe (tel que présenté par Levoy et Hanrahan [LH96]) ;
- Stocker les propriétés de surface de l'objet dans les échantillons. Ceci ne sera possible ici que pour des scènes synthétiques.

Ces deux approches seront explorées ici.

Il est important de noter que nous effectuerons notre rendu en utilisant la technique de ré-échantillonnage de Levoy et Hanrahan [LH96] plutôt que la technique d'apposition de textures de Gortler *et al.* [GGSC96] car bien que cette dernière soit plus rapide (elle utilise les calculs d'illumination du matériel), il ne serait pas possible de faire varier l'illumination à l'intérieur d'un même polygone texturé avec l'implémentation matérielle (la géométrie peut varier à l'intérieur d'un de ces polygones).

4.2.3 Modèle d'illumination

Le modèle d'illumination que nous utiliserons est similaire au modèle implémenté par OpenGL [ARB92] et partage des caractéristiques avec le modèle de Hall tel que décrit par Glassner [Gla89]. Nous avons choisi un modèle d'illumination locale car on envisage d'effectuer les calculs en temps réel. À cause de la complexité du calcul et des interréflexions, on ne peut pas utiliser un modèle d'illumination globale pour ré-illuminer en des temps interactifs.

Pour que l'illumination ajoutée soit cohérente avec celle qui est déjà présente dans le champ capturé, on doit utiliser le même modèle d'illumination pour la ré-illumination que celui qui a été utilisé lors de sa création. Pour des scènes réelles, tout dépendant de l'illumination présente,

on devrait typiquement utiliser un modèle d'illumination globale (c'est ce qui est généralement fait dans les travaux sur la réalité augmentée [FGR93] [NSD94] [DRB97]).

Le modèle d'illumination utilisé est exprimé par l'équation suivante :

$$I = I_a + \sum_{l=\text{lumières}} \left(k_a I_{a,l} + f_{\text{att},l} I_l \Big(k_d (\overline{N} \cdot \overline{L_l}) + k_s (\overline{N} \cdot \overline{H_l})^e \Big) \right) \tag{4.1}$$

où $f_{\text{att},l}$ est un facteur d'atténuation défini par $f_{\text{att},l} = \frac{1}{a_0 + a_1 d_l + a_2 d_l^2}$, et avec les variables

I	Intensité[7] calculée pour le pixel
I_a	Intensité ambiante globale
$f_{\text{att},l}$	Facteur d'atténuation pour la source l
$I_{a,l}$	Contribution ambiante de la source l
I_l	Contribution diffuse et spéculaire pour la source l
k_a, k_d, k_s	Composantes ambiante, diffuse et spéculaire de la surface
e	Coefficient de rugosité de la surface
$\overline{N}$	Vecteur normal à la surface au point d'intersection
$\overline{L_l}$	Vecteur du point d'intersection à la lumière
$\overline{H_l}$	Bissectrice entre le vecteur à l'oeil et le vecteur à la lumière
d_l	Distance du point d'intersection à la lumière l.

La dépendance de λ de I, I_a, k_a, k_d, k_s, $I_{a,l}$ et I_l a été explicitement ignorée dans cette équation. On applique séparément le modèle sur chacun des canaux RGB. Notons que le facteur d'atténuation f_{att} n'est pas physiquement correct (la diminution correcte pour une source ponctuelle peut être approximée en utilisant $a_0 = a_1 = 0$ et $a_2 = 1$). Notons aussi que $I_{a,l}$, les contributions ambiantes des sources de lumières, ont été inclues car elles faisaient partie du modèle d'illumination implémenté par OpenGL. Ce modèle simple a l'avantage d'être relativement peu coûteux à calculer, et suffisamment complexe pour simuler des reflets spéculaires décents.

[7] Pour être vraiment correct au niveau de la radiométrie, on devrait parler de radiance émise plutôt que d'intensité lumineuse. Une introduction et un traitement complet de la radiométrie n'est pas nécessaire ici et peut être retrouvé dans le livre de Cohen et Wallace [CW93b].

Modèle modifié

Pour notre application, on doit calculer et additionner la contribution des lumières ajoutées. Voici l'équation du modèle d'illumination modifié :

$$I = I_R + \sum_{l=\text{nouvelles lumières}} \left(\tilde{k}_a I_{a,l} + f_{\text{att},l} I_l \left(\tilde{k}_d (\overline{\tilde{N}} \cdot \overline{L_l}) + \tilde{k}_s (\overline{\tilde{N}} \cdot \overline{\tilde{H}_l})^e \right) \right) \tag{4.2}$$

où

I_R	Intensité stockée dans le champ de lumière
$\tilde{k}_a, \tilde{k}_d, \tilde{k}_s$	Composantes ambiante, diffuse et spéculaire *estimées* de la surface
$\overline{\tilde{N}}$	Vecteur normal *estimé* au point d'intersection

et où le point d'intersection où on fait le calcul est aussi estimé (ce calcul est décrit à la section 4.2.5).

L'estimation du point d'intersection, du vecteur normal, ainsi que des propriétés de surface font l'objet des sections 4.2.5, 4.2.7 et 4.2.6, respectivement. La calcul consiste simplement à ajouter les contributions des nouvelles lumières. Pour le cas où on stocke les propriétés de surface dans le champ de lumière, on utilise la formule 4.1, mais cependant avec le point d'intersection et la normale estimés.

4.2.4 Extension du champ de lumière

Comme vu à la section précédente, l'algorithme d'illumination adopté nécessite les paramètres suivants :

- La direction vers l'oeil (i.e. le rayon provenant de la caméra) ;
- Le point d'intersection de ce rayon avec la surface ;
- Les propriétés de surface au point d'intersection ;
- Le vecteur normal sur la surface au point d'intersection ;
- Les positions et descriptions des nouvelles lumières.

La position et l'orientation de la caméra sont toujours connues et fournissent ainsi le vecteur de direction vers l'oeil pour chaque rayon (on n'a bien sûr affaire qu'à des rayons primaires

ici). Les positions et descriptions des lumières ajoutées sont aussi toujours connues. On a encore besoin d'estimer le point d'intersection, les propriétés de surface et le vecteur normal. Nous avons étendu la structure de l'échantillon du champ de lumière de façon à contenir la distance à partir d'un des deux plans (le plan uv), ainsi que le vecteur normal pour le point d'intersection du rayon avec la surface (voir figure 4.2). Notons que la profondeur est signée, car le point d'intersection pourrait se situer devant, derrière ou entre les deux plans de la paramétrisation. On stocke tous ces paramètres lorsqu'on crée le champ de lumière étendu sur des scènes synthétiques. Notons que si l'objet représenté par champ de lumière subit des transformations, il faudra transformer inversement les valeurs stockées dans le champ de lumière au moment de faire les calculs d'illumination.

```
typedef struct {
    unsigned char r ; /* valeurs de radiance */
    unsigned char g ;
    unsigned char b ;
    float depth ;     /* profondeur */
    signed char nx ;  /* composantes du vecteur normal */
    signed char ny ;
    signed char nz ;
} EchantillonEtendu ;
```

FIG. 4.2: Structure d'un échantillon du champ de lumière étendu.

La taille des valeurs de radiance est la même que celle utilisée dans le travail de Levoy et Hanrahan [LH96]. Les normales sont normalisées. On a choisi de discrétiser celles-ci avec 128 valeurs négatives et 128 valeurs positives pour chaque composante (axe). La profondeur est stockée ici sous forme de nombre à point flottant, mais on pourrait aussi bien la stocker sous forme entière, comme pour le tampon de profondeur.

Nous utiliserons sélectivement les champs de cette structure dans nos techniques de rendu. De plus, les normales stockées pourront être utilisées pour caractériser les erreurs dues à l'approximation de ces dernières.

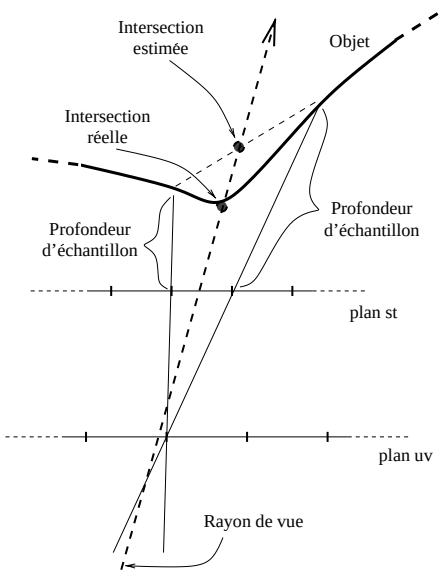


FIG. 4.3: Interpolation linéaire de la profondeur.

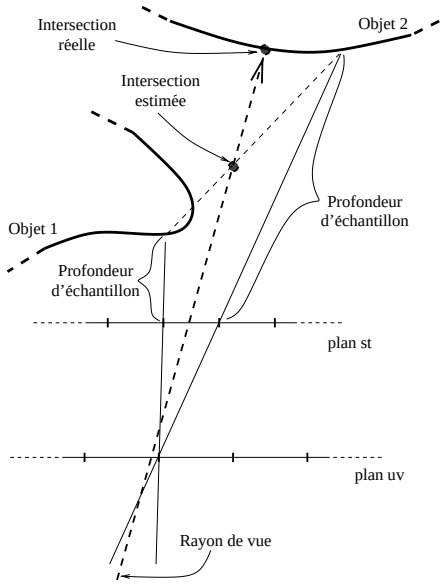


FIG. 4.4: Erreur due à l'interpolation.

4.2.5 Calcul du point d'intersection

En utilisant la profondeur échantillonnée

Si on utilise la profondeur stockée dans les échantillons du champ, on peut estimer le point d'intersection de la façon suivante : on extrapole dans l'espace les profondeurs des 4 ou 16 échantillons adjacents (les distances sont calculées selon la direction du rayon et à partir du plan uv) pour trouver les points d'intersection de ces derniers, et puis on effectue une interpolation bilinéaire ou quadrilinéaire sur ces distances (voir figure 4.3). Il est à noter que cette technique ne donne qu'une approximation linéaire aux points d'intersection. Ceci peut mener à des erreurs, notamment lorsque deux échantillons adjacents intersectent deux objets différents (voir figure 4.4).

Pour essayer de remédier à ce problème, on pourrait utiliser un seuillage pour disqualifier les échantillons adjacents dont la profondeur est trop différente de celle de ses voisins. Une méthode similaire est utilisée par Oliveira et Bishop [OB98] pour reconstruire des maillages partiels à partir de structures similaires. Nous n'avons pas intégré cette modification dans l'implémentation notre algorithme.

Sans utiliser la profondeur échantillonnée

On pourrait aussi estimer le point d'intersection sans utiliser de profondeur. On pourrait construire une approximation grossière de la forme de l'objet en utilisant les silhouettes présentes dans les images avec st fixe (en supposant qu'on puisse aisément segmenter l'arrière-plan de ces images en utilisant, par exemple, une couleur connue, ou encore un arrière-plan dont on a l'image) pour sculpter un *octree*, ce qui nous donnerait un modèle volumétrique de l'objet. Cette approche a été utilisée par Szeliski [Sze93] pour construire un modèle approximatif de la scène lors de la création de champ de lumière [GGSC96]. Ce modèle volumétrique est ensuite lissé en 3D en utilisant un algorithme de simplification géométrique [Tau95]. Le volume ainsi créé est un super-ensemble (*superset*) de l'enveloppe visuelle de l'objet et ne garantit pas une représentation correcte de la surface. Les cavités dans les objets ne seront pas détectées par cette méthode (on peut penser à l'intérieur d'une tasse, par exemple).

Aussi, on pourrait envisager faire des correspondances stéréo entre plusieurs images avec st fixe. Une technique qui puisse effectuer des correspondances sur de multiples images à la fois pourrait être utile ici. Le grand nombre d'images qu'on possède ici dans le cas du champ de lumière devrait nous permettre d'obtenir de bons résultats pour estimer la forme grossière de l'objet.

De plus, une approximation du modèle pourrait être calculée à l'avance à partir du champ de lumière ordinaire et conservée avec ce dernier. En plus de fournir une approximation du point d'intersection, cette approximation géométrique pourrait aussi être utilisée pour approximer les normales (voir section 4.2.7).

4.2.6 Calcul des propriétés de surface

Stockage direct à partir de scènes synthétiques

En créant nos propres champs de lumière à partir de rendus de scènes synthétiques, on peut stocker directement les propriétés de surface dans les échantillons. On stocke la couleur assignée à la surface au point d'intersection. On stocke seulement k_d pour les canaux R,G et B. Pour faire correctement, il faudrait stocker toutes les propriétés pour chaque échantillon, c'est-à-dire dans notre cas k_a, k_d et k_s pour chaque canal.

On procède au rendu par lancer de rayons, sans effectuer de calculs d'illumination. Ceci nous permet de générer des images avec des champs pour lesquels il n'y a aucune illumination fixe. Le champ de lumière ainsi créé ne contient cependant pas d'ombres projetées par l'objet sur lui-même.

À partir de champs de lumière

Pour utiliser les propriétés de surface des objets représentés par champ de lumière, on doit d'abord les récupérer. L'utilisation directe des valeurs de radiance présentes dans le champ serait erronée car ces valeurs sont le résultat de calculs d'illumination. Bien que les couleurs émises à partir d'un objet sont reliées à la couleur de la surface, cette couleur est fortement modulée par la couleur de la source de lumière. Le domaine de la vision offre malheureusement encore peu de ressources pour pouvoir récupérer les propriétés de surface d'objets à partir d'images. De récents résultats à ce sujet sont présentés dans Sato *et al.* [SWI97]. Peut-être serait-il possible d'obtenir de meilleurs résultats à partir d'un champ de lumière étant donné le grand nombre d'images disponibles. Nous n'explorerons pas cette avenue de recherche dans ce travail.

Si on ne veut qu'ajouter des reflets spéculaires sur la surface de l'objet, il est probablement suffisant d'assumer une couleur uniforme neutre pour l'objet. Nous avons implémenté ceci en illuminant les surfaces comme si elles étaient d'un gris neutre. Un problème avec cette méthode repose dans la contribution de l'illumination diffuse présente dans les calculs (les surfaces auront une apparence légèrement grisâtre). On pourrait aussi modifier le modèle d'illumination pour ne pas ajouter de composante diffuse au calcul, c'est-à-dire dans le but d'ajouter seulement des reflets spéculaires sur les objets.

4.2.7 Calcul du vecteur normal à la surface

Utilisation des normales échantillonnées

Pour calculer une approximation du vecteur normal à la surface, on peut utiliser les normales stockées dans la structure de l'échantillon étendu (bien qu'on veuille typiquement éviter cela, voir section 4.6). On utilise soit le vecteur de l'échantillon adjacent le plus près, ou encore

on fait une moyenne des vecteurs stockés dans les échantillons adjacents (soit en uv, en st, ou en uv et st), pondérée par la proximité des intersections par rapport aux coordonnées u, v, s, t du rayon.

Quand on utilise "l'interpolation" uv et st de cette manière, on obtient des normales qui varient de façon continue (on se trouve à enlever les variations rapides des normales sur la surface). Ceci a pour effet d'adoucir les variations de l'illumination sur l'objet. Une autre technique pour adoucir ces variations consisterait à effectuer l'interpolation *après* les calculs d'illumination (cette technique est élaborée en discussion).

Un traitement vraiment correct des normales consisterait à représenter et conserver une approximation de la distribution de ces normales pour chaque échantillon et d'utiliser cet ensemble de normales pour faire le calcul d'illumination [Fou92]. Cela revient à "filtrer" les normales. On peut visualiser l'importance du filtrage de normales en considérant leur effet sur l'illumination lorsqu'elles varient beaucoup à petite échelle, comme c'est le cas pour des surfaces exhibant des comportements anisotropiques. Il serait très coûteux de stocker ces distributions de normales pour chaque échantillon et nous n'avons donc pas implémenté cette technique ici.

Approximation du vecteur normal

Pour calculer une approximation du vecteur normal à la surface sans l'utilisation des normales échantillonnées, on utilise la position des points d'intersection adjacents à notre rayon (calculés comme à la section 4.2.5). À partir de quatre points d'intersection P_{00}, P_{01}, P_{10} et P_{11} dans l'espace, on utilise une technique très simple pour approximer la normale : on calcule le produit vectoriel normalisé des deux diagonales

$$\overline{N} = \frac{(P_{11} - P_{00}) \times (P_{01} - P_{10})}{||(P_{11} - P_{00}) \times (P_{01} - P_{10})||}$$

On pourrait utiliser plus d'échantillons voisins pour arriver à un meilleur estimé, ou encore calculer le plan qui passe le plus près des points et utiliser la normale de ce plan. Par contre, il faut faire bien attention de garder le coût de ce calcul assez bas, parce qu'il est effectué pour le calcul de ré-illumination de chaque pixel.

Une autre méthode consisterait à faire une triangulation des points d'intersection adjacents, et d'utiliser la moyenne des normales de ces deux triangles. Cette opération est souvent effectuée en tenant compte de l'aire de chaque triangle dans le moyennage. De plus, on doit utiliser les deux triangulations possibles des quatre points et moyenner les normales ainsi obtenues. Il faut garder à l'esprit qu'on effectue ces calculs pour chaque pixel de l'image produite au moment du rendu. Cette méthode serait trop coûteuse pour notre application.

Aussi, si on garde un modèle grossier de la scène, comme suggéré à la section 4.1.3, on pourrait intersecter le rayon avec ce modèle pour obtenir une normale. Par contre, cette intersection serait probablement trop coûteuse pour être effectuée en temps réel.

Il est important de noter que pour l'évaluation des normales, on devrait limiter l'utilisation des échantillons voisins à ceux qui sont sur la même surface que celle que l'on tente d'intersecter. Oliveira et Bishop [OB98] utilisent une heuristique sur les différences entre les profondeurs des échantillons adjacents pour déterminer si la surface est distincte ou non.

4.3 Taxinomie des techniques de ré-illumination

Les approches discutées dans les sections précédentes peuvent être combinées de différentes façons pour obtenir une méthode de ré-illumination de champ de lumière. Nous avons tenté d'en extraire les combinaisons les plus pertinentes et possibles. Elles sont illustrées au tableau 4.1 à la page 80, en ordre de complexité croissante. Par souci de clarté, nous avons assigné un numéro de catégorie à chaque combinaison, dont nous nous servirons pour référer à une méthode particulière dans le texte.

Lors de cette excursion, les méthodes A0, A1.1, A1.2, A2.1 et A2.2 ont été implémentées. La méthode A0 correspond au rendu par champ de lumière sans ré-illumination et la méthode A4 correspond au but ultime, c'est-à-dire au rendu par champ de lumière avec élimination de l'illumination présente dans le champ capturé ainsi qu'avec ré-illumination sans utilisation de profondeurs ou normales échantillonnées.

4.4 Implémentation

L'implémentation de ce logiciel a été effectuée en langage C++, et les temps de calcul donnés ont été mesurés sur un SGI avec processeur MIPS R10000 et 128 MB de mémoire vive. Nous avons inclu les capacités de ré-illumination dans le même système qui a servi à explorer le champ de visibilité (voir section 3.4).

4.5 Résultats

Dans cette section nous illustrons les résultats de l'application de notre algorithme sur des champs de lumière simples.

Nous avons créé un champ de lumière d'une sphère sans lumière, pour vérifier notre algorithme. La sphère utilisée dans les exemples n'est ni complètement spéculaire ni diffuse. Pour tester la validité de nos champs de lumière étendus, notre logiciel peut afficher sélectivement la couleur, la profondeur, les composantes x, y ou z de la normale, ou les composantes x, y ou z de l'intersection calculée, de façon similaire au rendu de l'objet (on peut naviguer interactivement autour de l'objet pour observer ces propriétés). La figure 4.5 contient un rendu d'une image de profondeur ainsi obtenu pour la sphère (Note : les hautes intensités représentent les surfaces près de l'observateur).

Pour cet exemple, nous avons utilisé une faible résolution de 15×15 sur le plan uv et 30×30 sur le plan st (le champ occupe donc 2.025 MB en mémoire). Nous avons rendu la sphère sans lumière, et avons stocké les propriétés de surface dans le champ au lieu de la radiance. Les figures 4.6, 4.7 et 4.8 montrent la sphère ré-illuminée avec une, deux et trois lumières, respectivement, *sans interpolation* (ce qui explique la pixelisation apparente, la résolution des images elles-mêmes est assez haute pour ne pas entrer en jeu ici). Les positions des lumières sont indiquées par les petites boîtes dans les images (elles sont placées à des profondeurs différentes). Ces rendus correspondent à la catégorie A1.1 (voir tableau 4.1 à la page 80).

Un exemple plus intéressant est illustré à la figure 4.9. On y voit un champ de lumière

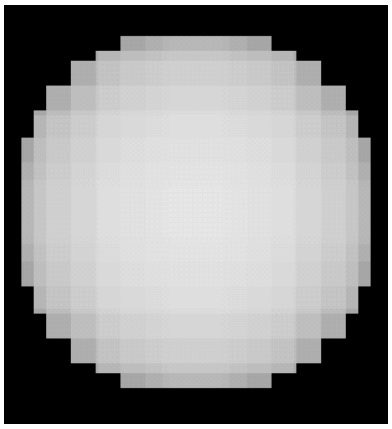


FIG. 4.5: Rendu du champ de profondeur pour la sphère.

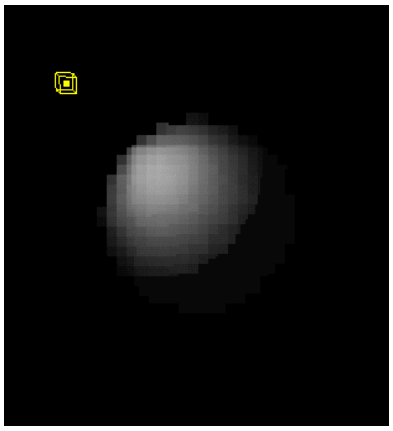


FIG. 4.6: Sphère rendue avec une nouvelle lumière.

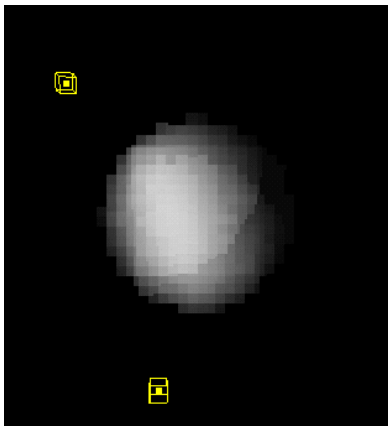


FIG. 4.7: Sphère rendue avec deux nouvelles lumières.

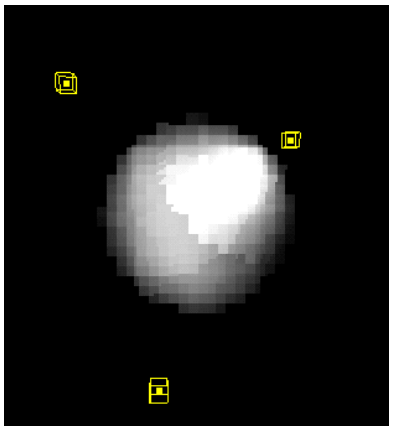


FIG. 4.8: Sphère rendue avec trois nouvelles lumières.

d'un modèle de noeud mathématique produit par *KnotPlot*[8], sans ré-illumination. La scène originale contenait de multiples lumières (environ une dizaine) et fut capturée ainsi (on a stocké la radiance dans le champ de lumière). Les résolutions sont de 24×24 sur le plan uv et de 48×48 sur le plan st (ce qui est assez peu pour un rendu décent tel qu'illustré par les "gros" pixels de l'image, mais suffisant pour tester notre algorithme).

À la figure 4.11, on peut voir la même scène après qu'on ait ajouté une lumière, avec la ré-illumination utilisant les normales échantillonnées (catégorie A1.2). Pour évaluer l'effet

[8]Gracieuseté de Robert Scharein, du laboratoire Imager à UBC.

de l'approximation des normales, il faut comparer cette image avec celle de la figure 4.13 (catégorie A2.2). Les figures 4.10, 4.12 et 4.14 montrent les mêmes images[9] mais qui utilisent cette fois l'interpolation de la radiance, des points d'intersection et des normales en uv et st. Il est à noter que bien qu'on puisse se passer d'utiliser les normales échantillonnées, toutes ces images sont calculées en utilisant les profondeurs échantillonnées (on en a besoin). Aussi, on travaille avec des résolutions assez basses car la taille des structures de données en mémoire est considérable.

4.5.1 Problèmes aux silhouettes

Des artefacts près des silhouettes des objets sur les images ré-illuminées avec interpolation $uv + st$ sont clairement visibles aux figures 4.12 et 4.14. Des effets similaires apparaissent lorsqu'on utilise l'interpolation sur seulement uv ou seulement st. Ces effets sont dus au fait qu'on utilise les échantillons vides adjacents à l'objet (lorsqu'il n'y a pas d'intersection pour ces échantillons, le vecteur normal est indéfini et la profondeur infinie).

On pourrait éliminer ces contributions fautives en faisant un seuillage sur la différence entre les profondeurs des échantillons, pour éviter de se servir des échantillons vides. Cela aiderait grandement à réduire ces artefacts, bien qu'on verrait apparaître des discontinuités là où le ré-échantillonnage change de nombre de vecteurs utilisés pour effectuer l'approximation.

Une autre technique consisterait à effectuer l'interpolation après le calcul d'illumination. Les artefacts sur les silhouettes seraient alors plus flous.

4.5.2 Temps de rendu

Pour nous permettre d'estimer le coût additionnel de notre algorithme, nous avons mesuré les temps de rendu pour les catégories A1.2 et A2.2, en augmentant le nombre de nouvelles sources de lumière. Les résultats sont affichés à la figure 4.15. Comme prévu, nous observons que l'approximation des normales résulte en un coût constant, peu importe le nombre de nouvelles sources de lumière. De plus, le temps de rendu augmente de façon linéaire avec le nombre de sources de lumière.

[9] Noter qu'on a changé la position de la nouvelle source de lumière pour les images où on utilise l'interpolation.

FIG. 4.9: Noeud rendu sans ré-illumination (aucune interpolation).

FIG. 4.10: Sans ré-illumination (interpolation $uv + st$).

FIG. 4.11: Avec ré-illumination (aucune interpolation, normales échantillonnées).

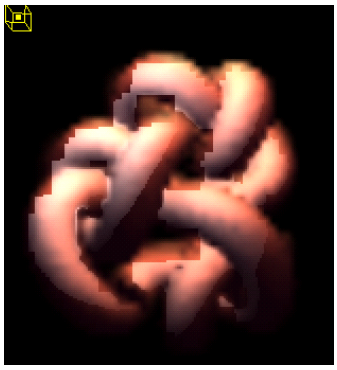


FIG. 4.12: Avec ré-illumination (interpolation $uv + st$, normales échantillonnées).

FIG. 4.13: Avec ré-illumination (aucune interpolation, normales approximées).

FIG. 4.14: Avec ré-illumination (interpolation $uv + st$, normales approximées).

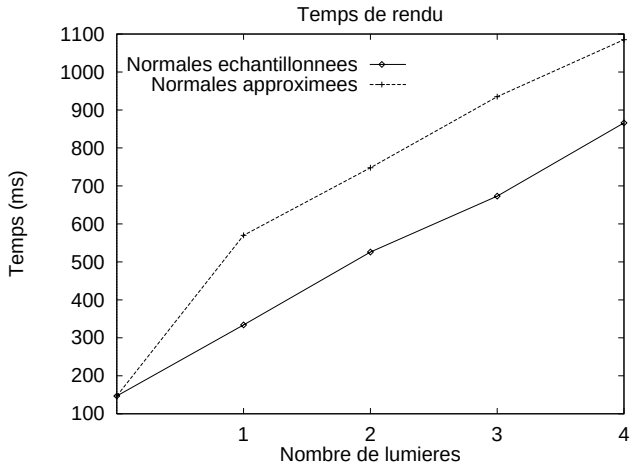


FIG. 4.15: Temps de rendu pour la scène du noeud (échantillonnage sans interpolation).

4.6 Discussion

Les résultats présentés à la section précédente sont quelque peu préliminaires. Il reste encore beaucoup de travail à faire pour explorer en profondeur le potentiel de la ré-illumination de champ de lumière.

Si les champs de lumière doivent être utilisés dans des applications réelles, leur ré-illumination est cruciale au succès de leur intégration dans des scènes réelles. Il faut quand même se poser des questions de fond. Bien que la ré-illumination de ces objets soit vivement nécessaire, le stockage de la profondeur dans les échantillons du champ contribue à accroître la taille des structures de données déjà très lourdes. Après tout, le stockage de tous ces échantillons de profondeur revient un peu à stocker la géométrie, car on pourrait reconstruire une approximation grossière de l'objet à partir de ses profondeurs. Pour plusieurs modèles actuels, la géométrie pourrait être stockée plus efficacement dans un maillage ou encore dans un texel volumétrique, tout dépendant de l'échelle à laquelle on désire faire le rendu. C'est en quelque sorte l'approche du rendu basé sur les vues présenté à la section 2.5 : un rendu par modèles simplifiés qui projette une texture capturée du monde réel, ce qui nous permet de changer l'illumination (bien que de façon limitée).

Néanmoins, le rendu par champ de lumière offre des avantages bien clairs comparés au rendu traditionnel par modèles. Des effets d'illumination complexes peuvent y être capturés,

sans limite sur la complexité de ces derniers (la limite étant la fréquence d'échantillonnage avec laquelle on peut les capturer).

De plus, au fur et à mesure que les numériseurs de profondeur (*range scanner*) deviennent plus communs et moins coûteux, la capture de champs de lumière d'objets réels avec radiance et profondeur deviendra plus facile, et probablement une extension acceptable pour permettre la ré-illumination. Quelques projets de recherche récents [Lev98] [RBMT98] témoignent de cette tendance.

Mais il ne faut pas se leurrer : pour les nouvelles sources de lumière introduites, on ne peut simuler que les effets de rendu qui sont possibles de simuler dans les systèmes courants de rendu par modèles (e.g. par tracer de rayons, radiosité, etc.), avec les limites inhérentes à ces méthodes.

Il serait difficile de capturer la transparence présente dans les objets réels. L'utilisation d'arrière-plans connus et calibrés serait possible pour déterminer la couleur d'un objet (bien que cette technique ne soit pas sans problèmes, voir Smith [SB96] à ce sujet), mais les positions des surfaces seraient plus difficiles à évaluer. On peut penser, par exemple, au problème de trouver une estimation de la position spatiale pour des cheveux, ou encore au problème de trouver une estimation de la normale pour des cheveux (en fait, il existe là plutôt une *distribution* de normales). De plus, en considérant plusieurs arrière-plans différents pour obtenir la quantité de transparence sur une surface, on aura des erreurs additionnelles dues aux variations de l'interaction de la lumière à de longueurs d'ondes différentes lors de la réfraction sur les surfaces transparentes.

Aussi, la simulation des réflexions ou réfractions sur l'objet pourrait causer des problèmes. Dans le cas d'objets hautement réflectifs, cette composante de l'apparence de l'objet peut jouer un rôle assez important, et les erreurs dues à l'échantillonnage des positions et des normales pourraient être amplifiées dans les couleurs réfléchies par l'objet. Par exemple, les discontinuités dues aux normales seraient beaucoup plus évidentes sur une surface mirroir représentée par un champ de lumière.

De plus, d'autres problèmes se posent par rapport à l'utilisation même de la structure du champ de lumière échantillonné. Même avec un préfiltrage correct, un effet important dû à

un changement d'occlusion d'un échantillon à un autre ne pourrait être interpolé correctement. Aussi, un effet qui produit des variations à hautes fréquences en intensité lumineuse ne pourrait pas être capturé ni reproduit correctement.

4.7 Travaux futurs

Il y a assez de travaux à faire reliés à la ré-illumination de champ de lumière pour faire un autre mémoire. Nous décrivons ici quelques avenues potentiellement intéressantes à explorer.

4.7.1 Quantification des erreurs d'approximation

Il reste encore à faire une analyse pour nous permettre d'estimer la quantité d'erreurs introduites en approximant les normales avec les points d'intersection adjacents. Pour des scènes synthétiques, on a à notre disposition le modèle géométrique pour nous fournir les valeurs espérées. On pourrait conduire des expériences pour évaluer les erreurs introduites par nos approximations en comparant avec le modèle exact. Le rendu par champ de lumière est de par sa nature une approximation, et les résultats d'une telle analyse pourraient nous aider à déterminer de meilleures résolutions et paramètres pour effectuer la création de ces champs, possiblement avec des bornes d'erreur. De plus, étant donnée une estimation de l'erreur, on pourrait s'en servir comme critère de suréchantillonnage ou encore plus guider la création d'une structure adaptative.

4.7.2 Traitement des silhouettes

Nous pourrions aussi traiter plus en profondeur les problèmes d'interpolation aux silhouettes des objets. Bien que cela améliorerait le rendu, il ne suffit pas de se débarrasser des échantillons adjacents jugés comme faisant partie d'une surface distincte. Si on utilise cette méthode de rejet, un échantillon près de la silhouette utilisera moins d'échantillons voisins pour effectuer son interpolation et il en résultera une discontinuité C^1 entre les zones de l'image là où l'on change de paramètre. Cette discontinuité s'apparente à celle qui se présente avec la technique d'illumination de Gouraud [Gou71].

Bien que cette méthode de rejet fonctionnerait généralement bien, les silhouettes intérieures à l'objet, c'est-à-dire les silhouettes dues à des replis d'un objet sur lui-même ou encore des silhouettes entre deux objets distincts seraient beaucoup plus difficiles à traiter. Mark *et al.* [MMB97] utilisent un terme de "confiance" pour juger la connectivité des surfaces dans leur algorithme de reprojection. On pourrait utiliser un terme similaire pour faire le rejet d'échantillons appartenant à des surfaces distinctes.

4.7.3 Calculs de visibilité

L'utilisation d'une structure d'échantillon étendue comme celle présentée ici tend naturellement au développement de nouvelles techniques pour calculer la visibilité relative entre un objet représenté sous forme de champ de lumière étendu et un autre objet. Comme on peut obtenir un estimé de l'enveloppe des objets, on pourrait déterminer lequel se trouve le plus près de l'observateur, pour un rayon donné. On pourrait accessoirement utiliser cette même technique pour joindre deux objets représentés par champ de lumière en une seule représentation [Chi98]. À cet effet, il serait intéressant d'explorer la qualité du calcul de visibilité relative à un objet représenté par champ de lumière étendu sans l'utilisation d'une approximation grossière de sa forme.

Plusieurs problèmes peuvent se présenter en ce qui concerne la détermination de la forme d'un objet pour son calcul de visibilité. Un champ de lumière pourrait être mal aligné car la calibration de caméras réelles peut comporter des imprécisions. Aussi, des modèles approximatifs pourraient simplement s'avérer trop grossiers pour les utiliser pour cette tâche. Dans le cas où on aurait des transparences, il pourrait survenir des cas où on aurait besoin de plusieurs couches de profondeur pour calculer correctement la visibilité.

4.7.4 Ombres

Dans notre travail, nous avons considéré l'effet de la scène sur un objet représenté par champ de lumière inséré dans celle-ci, et non l'effet inverse. Plus particulièrement, un objet par champ de lumière pourrait projeter des ombres sur le reste de la scène. Si la technique de rendu pour la scène utilise le tracer de rayons, il est alors très rapide d'obtenir l'opacité de l'objet

par champ de lumière en supposant qu'on puisse stocker un canal alpha dans chacun de ses échantillons. Cela consiste simplement à accéder à un échantillon (ou 4, ou 16, tout dépendant de l'interpolation choisie) du champ de lumière. On pourrait automatiquement projeter des ombres floues par l'interpolation du canal alpha de l'objet représenté par champ de lumière.

D'autres méthodes d'ombrage nécessitent une description de l'objet sous forme de maillage. Une reconstruction grossière à partir d'un champ de lumière est possible, et probablement suffisante pour la tâche de projeter des ombres. L'utilisation d'une reconstruction grossière serait probablement plus appropriée si la distance à la lumière est grande. L'utilisation d'un canal alpha reste toutefois plus général et intéressant.

Bien que les calculs d'illumination sur le champ de lumière fonctionnent correctement lorsque celle-ci se trouve à l'intérieur de l'objet (i.e. si la lumière se trouve derrière une surface, aucune illumination ne sera ajoutée et ce, à cause du vecteur normal), on ne peut pas projeter sur la scène les ombres dues à cette lumière. On doit donc restreindre la position de la lumière à la région située en dehors de l'enveloppe convexe de l'objet.

4.7.5 Utilisation de valeurs physiques

La tendance courante dans le domaine du rendu avec illumination globale est à l'utilisation de valeurs physiques de radiance et de radiosité pour la simulation de la propagation de la lumière. Les intensités lumineuses perçues par le système visuel humain varient par environ 9 ordres de magnitude. On voudrait utiliser une représentation de la couleur qui tient en compte ces larges variations.

Cette même idée s'applique aux images : on peut stocker des images de radiance (*radiance map*), par exemple, pour capturer "l'image" d'un environnement autour d'un point. Récemment, une méthode a été proposée pour permettre la création d'images de radiance de scènes réelles à partir de photographies [DM97].

Dans la représentation par champ de lumière, au lieu de stocker nos échantillons de radiance en termes de trois canaux RGB de 8 bits qui ne peuvent représenter que deux ordres de magnitude, on pourrait envisager de stocker la radiance sous une forme permettant plus de variation. L'utilisation de nombres réels est prohibitive à cause de leur taille, mais des représentations

plus compactes mieux adaptées aux facteurs de perception du système visuel humain existent [WL97a] [WL97b].

4.7.6 Transformations sur les objets

Dans la plupart des applications de recherche sur le champ de lumière, les objets représentés sont fixes dans l'espace (et le temps), et on permet à l'observateur de se déplacer autour de l'objet. On pourrait aisément déplacer la paramétrisation (les deux plans) dans l'espace pour déplacer l'objet correspondant, et il serait intéressant d'explorer comment on pourrait effectuer des déformations sur la paramétrisation pour obtenir des déformations de l'objet représenté. Toute déformation applicable par tracer de rayons (affine, déformation *free-form*, etc.) pourrait s'appliquer aux champs de lumière. Il ne suffit que de garder la déformation inverse pour garder le champ dans sa forme canonique. L'illumination doit toujours s'effectuer dans le référentiel de la scène, comme c'est le cas lors du tracer de rayons standard.

Si on arrive à effectuer une déformation correcte des champs de lumière, il serait possible d'utiliser ces derniers de façon analogue aux *texels* volumétriques sur les surfaces [Ney95b] [Ney95a], en assumant qu'on puisse les ré-illuminer. Bien que cela ne représente pas une technique d'animation très versatile pour de gros champs de lumière, cela pourrait nous permettre de faire une animation limitée de ce type d'objets. On pourrait rentabiliser le coût important en mémoire de cette représentation en répliquant des instances de ces *texels* par champ de lumière sur une surface (c'est-à-dire par l'utilisation de pointeurs multiples à un champ de lumière unitaire comme bloc de base).

4.7.7 Ombrage de l'objet sur lui-même

Pour tenir compte des ombres projetées par l'objet sur lui-même, on pourrait calculer le point d'intersection du rayon depuis la source de lumière, et en comparant les points calculés pour la caméra et pour la lumière, on pourrait décider d'ajouter ou non la contribution de la source. À cause des approximations impliquées sur le calcul de profondeur, on aurait probablement des problèmes dans les cas limites (e.g. ombre à angle rasant sur la surface).

4.7.8 Distributions de normales

Au lieu de stocker les vecteurs normaux dans chaque échantillon, on pourrait stocker une distribution de normales [Fou92] [Ney95b] et de cette façon diminuer les artefacts d'illumination dus à l'interpolation ou l'approximation de ces dernières.

4.7.9 Récupération des propriétés de surface des objets

Pour effectuer la ré-illumination d'un objet par champ de lumière, on doit connaître ses propriétés de surface. Étant donnés une simple image, ses paramètres d'illumination (positions, intensités et couleurs des lumières), la calibration de la caméra et un modèle géométrique grossier de la scène, le problème d'extraire les propriétés de surface est encore difficile (voir en section 4.1). De plus, sans les paramètres d'illumination, on doit utiliser des heuristiques qui potentiellement introduiront beaucoup d'erreurs (par exemple, les images texturées offrent toujours des difficultés toutes particulières).

Étant donné le champ de lumière d'un objet, nous avons beaucoup plus d'information à notre disposition qu'une seule image. Ce champ peut être considéré comme un grand ensemble d'images des mêmes surfaces prises à intervalles réguliers avec des angles différents. C'est encore mieux qu'une séquence vidéo où les images sont beaucoup moins organisées. Il serait intéressant d'explorer jusqu'à quel point on pourrait récupérer les propriétés de surface d'un objet en combinant toute l'information contenue dans les tranches de champ de lumière.

Une application directe de cette idée serait de tenter de récupérer la BRDF d'une surface à partir du champ de lumière. Étant donné qu'un point particulier d'une surface est visible dans plusieurs images (on peut obtenir un tel point d'intersection à partir de la profondeur stockée dans les échantillons), on peut trouver l'ensemble des rayons qui intersectent près de ce point, et ré-échantillonner le champ dans "l'autre sens" pour obtenir les valeurs de radiance qui quittent ce point (voir figure 4.16). Chaque échantillon fournit alors une valeur pour un θ_o, ϕ_o pour un point de surface et pour l'illumination courante. Aussi, si chaque image contient plus qu'un échantillon de la même surface dont les propriétés sont uniformes (i.e. la surface occupe plus qu'un pixel dans une image avec st fixe), pour plusieurs points différents sur la surface on a une variation de la position relative des lumières (θ_i, ϕ_i) et une variation de la position

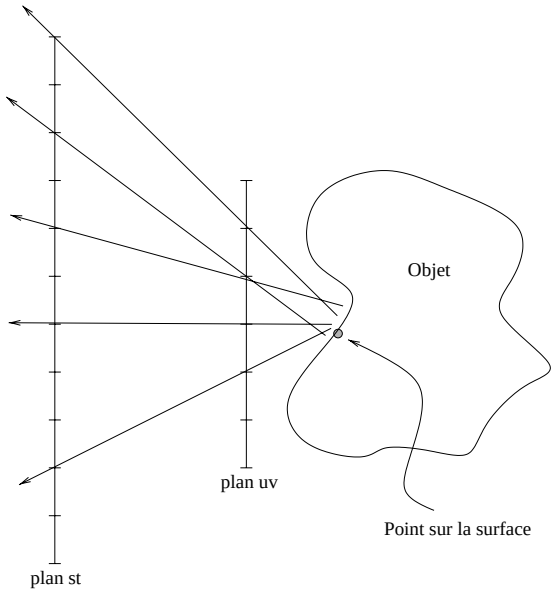


FIG. 4.16: Échantillons qui pourraient être utilisés pour calculer la BRDF à un point de surface.

relative de l'oeil combinées. On doit donc remplir un tableau de BRDF à partir des échantillons disponibles. Le problème difficile à résoudre est de retrouver les valeurs de BRDF pour une chaque paire d'angles incidents et sortants ($\theta_i, \phi_i, \theta_o, \phi_o$), car les valeurs de radiance stockées dans le champ de lumière sont le résultat d'une intégration de toute la lumière incidente sur la surface (l'irradiance, possiblement inconnue, surtout si la contribution des réflexions diffuses sur d'autres parties de l'objet ou de la scène est importante). Même en ne considérant que l'illumination directe de quelques sources de lumière, il serait difficile de séparer la contribution provenant de chaque θ_i, ϕ_i. Aussi, si on ne connaît pas l'intensité des sources de lumières, on doit assumer une intensité et on obtient alors une BRDF à un facteur d'échelle près seulement.

4.7.10 Correspondances entre plusieurs images

À la façon d'un algorithme de correspondance stéréo, il serait intéressant de tenter d'exploiter la cohérence entre les images pour reconstruire un modèle tridimensionnel. Comme mentionné à la section précédente, dans le contexte du champ de lumière on possède une multitude d'images d'une scène espacées régulièrement sur un plan. Il est probablement possible d'obtenir de meilleurs résultats en utilisant toutes ces images plutôt que seulement une paire

d'images, comme c'est le cas pour la plupart des algorithmes de correspondance stéréo standards. Il serait intéressant de modifier un tel algorithme pour lui permettre d'utiliser toutes ces images simultanément pour produire une carte de disparité, pour tenter d'obtenir des résultats de haute qualité.

4.7.11 Élimination des ombres

Il est important de mentionner la nécessité d'enlever les ombres dans les champs de lumière. Si on désire *changer* l'illumination d'objets représentés par champ de lumière, les ombres changent, et doivent être soit enlevées ou recréées. Tout comme les techniques de détection de reflets spéculaires, la détection et l'élimination automatique des ombres est une tâche difficile [Ger87]. Comme pour les propriétés de surface, il serait peut-être possible d'utiliser la cohérence accrue et la redondance d'une plus grande quantité d'images disponibles pour éliminer les ombres.

4.7.12 Intégration avec d'autres algorithmes de rendu

Selon la méthode de rendu utilisée, on peut envisager différentes méthodes pour incorporer des objets représentés par champ de lumière dans la scène, et on pourrait parfois tirer avantage de certaines propriétés de la représentation par champ de lumière.

Par exemple, si on calcule une solution de radiosité indépendante du point de vue, on doit faire une intégrale sur la surface projetée de l'objet sur chaque élément de subdivision (*patch*) pour calculer le facteur de forme. On peut effectuer ce calcul rapidement sur un objet représenté par champ de lumière, car pour chaque rayon on ne doit faire qu'un simple accès à notre tableau (ré-échantillonnage). Avec notre structure étendue, on obtient la distance d'intersection et la normale, et on peut ensuite calculer le facteur de forme différentiel. En utilisant une méthode d'évaluation qui exploite la cohérence en espace image de façon similaire à celle utilisée par Gortler *et al.* [GGSC96] (décrite à la section 2.4.3), on pourrait estimer plus rapidement le facteur de forme.

L'intégration avec un système de tracer de rayons est triviale : avec notre représentation, chaque objet peut interagir avec un rayon comme avec n'importe quel autre type d'objet. Par

contre, les imprécisions sur les normales et les points d'intersection pourraient causer des imprécisions dans les réflexions sur l'objet représenté par champ de lumière. L'intégration dans un algorithme par tampon de profondeur ou par balayage de lignes serait aussi possible. Il suffirait d'ajouter un calcul d'illumination pour chaque pixel rendu dans le tampon de profondeur.

4.7.13 Représentations compactes

Si le champ de lumière est représenté sous forme hiérarchique, ou encore en décomposition par ondelettes, il serait intéressant d'explorer comment on pourrait intégrer plus rapidement le facteur de forme ou l'intensité de l'objet par rapport à un point [Lal98].

La représentation par champ de lumière occupe quatre dimensions, tout comme la plupart des représentations de BRDFs existantes, et une représentation par ondelettes de BRDFs a déjà été développée dans ce contexte [LF97]. La cohérence directionnelle des BRDFs a été exploitée [Rus98] et offre un degré de compression additionnel important. Il serait donc indiqué de tenter d'utiliser des paramétrisations différentes combinées avec un encodage par ondelettes du champ, pour voir si on peut réduire sa taille de façon importante en exploitant la cohérence dans ce champ.

4.7.14 Représentation de luminaires

La représentation de luminaires complexes est très importante pour la simulation physique correcte de l'illumination globale [Ash93]. Néanmoins, pour stocker la distribution de lumière émanant d'un luminaire, il faut stocker l'émission de radiance par rapport à la position sur le luminaire et par rapport à la direction émise. On pourrait tenter de stocker cette description avec une représentation par champ de lumière. Tout récemment, on a vu apparaître une méthode qui utilise cette approche [HKSS98]. Même si une telle représentation est plus flexible qu'une représentation goniométrique, elle reste quand même toujours victime de la résolution en $uvst$. Plus on projettera l'illumination du luminaire sur une surface éloignée, plus on verra apparaître la résolution de la paramétrisation choisie.

4.7.15 Représentation hiérarchique

Il serait intéressant d'explorer une représentation hiérarchique du champ de lumière dans deux buts : pouvoir allouer plus de détails dans les régions où il y a des variations importantes (e.g. des discontinuités, ou de hautes fréquences), et pour tenter de représenter de façon plus compacte le champ. L'allocation de "sous-champs" de manière hiérarchique à l'intérieur du champ de lumière de façon à nous permettre d'aller chercher plus de détails là où on en a besoin pourrait nous permettre d'améliorer grandement la qualité de rendu qu'on obtient avec cette méthode car la résolution limitée du champ est responsable de la plupart des artefacts observés au rendu et à la ré-illumination de ce dernier.

# de référence	Utilise la radiance ou les propriétés de surface échantillonnées	Utilise la profondeur échantillonnée	Utilise la normale échantillonnée	Élimine l'illumination présente	Propriétés de surface : Fixes (constantes)	Propriétés de surface : Spécifiées lors du rendu (scènes synthétiques seulement)	Propriétés de surface : Récupérées à partir des images	Intersection : Interpole les profondeurs échantillonnées	Intersection : Calcule à partir d'un modèle approximatif	Normale : Interpole les normales échantillonnées	Normale : Approxime à partir des profondeurs échantillonnées	Commentaires
A0	Oui	Non	Non	Non								Pas de ré-illumination (rendu par champ de lumière original)
A1.1	Oui	Oui	Oui	Non	✘			✘		✘		Ajout de nouvelles sources seulement
A1.2	Oui	Oui	Oui	N/A		✘		✘		✘		Toutes nouvelles sources
A1.3	Oui	Oui	Oui	Oui			✘	✘		✘		Toutes nouvelles sources
A2.1	Oui	Oui	Non	Non	✘			✘			✘	Ajout de nouvelles sources seulement
A2.2	Oui	Oui	Non	N/A		✘		✘			✘	Toutes nouvelles sources
A2.3	Oui	Oui	Non	Oui			✘	✘			✘	Toutes nouvelles sources
A3.1	Oui	Non	Non	Non	✘				✘		✘	Ajout de nouvelles sources seulement
A3.2	Oui	Non	Non	N/A		✘			✘		✘	Toutes nouvelles sources
A3.3	Oui	Non	Non	Oui			✘		✘		✘	But ultime

TAB. 4.1: Différentes approches au rendu par champ de lumière étendu.

Chapitre 5

Conclusion

"In 10 years, all rendering will be volume rendering."

—Jim Kajiya, *SIGGRAPH 1991*

De la même façon que le rendu volumétrique consiste en une méthode de force brute pour la modélisation (un échantillonnage discret du modèle dans l'espace), le rendu basé sur des images consiste en une méthode de force brute pour effectuer le rendu. Tout comme le rendu volumétrique peut représenter n'importe quel modèle, la représentation par champ de lumière peut exprimer tout effet de rendu désiré. On peut le considérer comme une méta-primitive de rendu [Lev97].

Le précalcul des champs de lumière, la cohérence de l'accès aux données de celui-ci ainsi qu'une complexité de rendu indépendante de la complexité de la scène offrent des avantages indéniables pour les applications exigeant du temps réel. Il y a un compromis entre la qualité de rendu désiré, le temps de calcul alloué et la quantité de mémoire disponible. La progression rapide des vitesses des *CPUs* et de la quantité de mémoire disponible offre l'espoir qu'on pourra obtenir de meilleures images en moins de temps dans le futur.

Est-ce que dans dix ans, tout le rendu consistera en du rendu par champ de lumière ? Nous émettons de sérieux doutes. On a atteint certaines limites inhérentes à la manipulation de représentations volumétriques en modélisation. Les problèmes émergeant en rendu par champ de lumière laissent présager une difficulté analogue. D'un autre côté, l'histoire nous a montré que des technique d'échantillonnage ont souvent pris le dessus sur des méthodes plus élégantes.

Par exemple, on peut penser à l'avènement des dispositifs d'affichage par pixel (*raster graphics*) plutôt que de dispositifs vectoriels (*vector graphics*), ou encore à l'utilisation répandue du tampon de profondeur pour effectuer l'élimination des surfaces cachées.

Les domaines de l'infographie et de la vision sont présentement submergés par l'engouement au développement de nouvelles techniques pour tenter d'étendre le rendu par champ de lumière. Une multitude d'idées y sont présentées pour diverses applications pratiques, mais le problème à résoudre est encore mal défini[1]. On tente de faire du rendu photoréaliste. Des échantillons capturés du monde réel sont souvent plus réalistes que ceux produits par nos algorithmes, mais c'est ce qui se passe "entre" ces échantillons qui est difficile de contrôler de façon exacte. Jusqu'à quel point ces nouvelles structures nous aident-elles à avancer vers ce but ?

Néanmoins, les techniques d'échantillonnage de la radiance présentées restent un *outil* potentiellement intéressant dans le cadre de certaines applications de l'infographie. Par exemple, il a déjà été démontré comment l'interpolation de vues peut servir à réduire les délais dans les systèmes de réalité virtuelle [MMB97]. On trouve un autre exemple dans le système de reprojection *Quicktime VR* [Che95], qui est déjà commercialisé.

Hormis la taille des données à stocker, le plus grand obstacle à l'utilisation répandue du champ de lumière comme technique de représentation reste, à notre avis, le fait que nous ayons à traiter d'un échantillonnage discret de l'illumination, et donc des limites inhérentes au processus d'échantillonnage. Nous avons espoir que les vastes expériences en ce domaine en infographie et en vision par ordinateur pourront aider à réduire les problèmes reliés à cette discrétisation, sans pour autant nécessairement garantir l'exactitude des résultats produits.

Dans ce mémoire nous avons présenté deux techniques différentes. Premièrement, nous avons présenté une exploration au précalcul de visibilité par échantillonnage dans le domaine des lignes dans l'espace, où on utilise une paramétrisation similaire à celle utilisée en rendu par champ de lumière. Deuxièmement, nous avons exploré une méthode d'extension de la technique de rendu par champ de lumière pour permettre de changer l'illumination de ces derniers.

[1] De façon similaire, les problèmes présentés en vision et en infographie sont eux aussi souvent mal définis [Ste95].

Bien qu'à la base ces techniques fonctionnent, elles souffrent toutes deux des aléas dus à l'échantillonnage discret. La solution obtenue en utilisant le champ de visibilité n'offre donc pas la garantie d'être exacte. En ce qui concerne la ré-illumination de champs de lumière, de nombreux travaux futurs pourraient être développés, tels que décrits à la section 4.7, et nous estimons qu'il y a une bonne possibilité d'obtenir de nouveaux résultats de recherche dans cette direction.

Les exemples donnés dans ce mémoire illustrent à la fois les possibilités et les difficultés de cette discrétisation dans l'espace. Les solutions adoptées et les suggestions d'améliorations ne sont que quelques idées parmi la grande panoplie de structures de représentation pour le rendu de scènes. Notamment, dans le spectre de ces techniques de représentation, on a décrit ici un ensemble de méthodes où on opte pour une représentation simple mais détaillée des données pour le rendu par opposition à des représentations plus compactes, qui utilisent des modèles empiriques potentiellement plus complexes.

Bibliographie

[AB91] Edward H. Adelson et J. R. Bergen. The Plenoptic Function and the Elements of Early Vision. Dans *Computational Models of Visual Processing*, édité par Michael Landy et J. Anthony Movshon, chapitre 1. MIT Press, 1991.

[App68] Arthur Appel. Some Techniques for Shading Machine Renderings of Solids. Dans *AFIPS 1968 Spring Joint Computer Conf.*, volume 32, pages 37–45, 1968.

[ARB90] John M. Airey, John H. Rohlf et Frederick P. Brooks, Jr. Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments. Dans *Proceedings of the Symposium on Interactive 3D Graphics*, édité par Rich Riesenfeld et Carlo Sequin, volume 24, pages 41–50, mars 1990.

[ARB92] OpenGL Architecture Review Board. *OpenGL Reference Manual, Release 1*, 1992.

[Ash93] Ian Ashdown. Near-Field Photometry : A New Approach. *Journal of the Illuminating Engineering Society*, 22(1) :163–180, hiver 1993.

[BK70] W. Jack Bouknight et K. C. Kelly. An Algorithm for Producing Half-tone Computer Graphics Presentations with Shadows and Movable Light Sources. Dans *Proc. AFIPS JSCC*, volume 36, pages 1–10, 1970.

[BK96] Peter N. Belhumeur et David J. Kriegman. What is the Set of Images of an Object under All Possible Lighting Conditions ? Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, juin 1996.

[Bli78] James F. Blinn. Simulation of Wrinkled Surfaces. Dans *Computer Graphics (SIGGRAPH '78 Proceedings)*, volume 12, pages 286–292, août 1978.

[BN76] James F. Blinn et Martin E. Newell. Texture and Reflection in Computer Generated Images. *Communications of the ACM*, 19(10) :542–547, octobre 1976.

[BN92] Thaddeus Beier et Shawn Neely. Feature-based Image Metamorphosis. Dans *Computer Graphics (SIGGRAPH '92 Proceedings)*, édité par Edwin E. Catmull, volume 26, pages 35–42, juillet 1992.

[Bou70] W. Jack Bouknight. A Procedure for Generation of Three-Dimensional Half-Toned Computer Graphics Presentations. *Communications of the ACM*, 13(9) :527–536, septembre 1970.

[Cat74] Edwin E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. Thèse de doctorat, Dept. of CS, U. of Utah, décembre 1974.

[Che95] Shenchang Eric Chen. Quicktime VR - An Image-Based Approach to Virtual Environment Navigation. Dans *SIGGRAPH 95 Conference Proceedings*, édité par Robert Cook, Annual Conference Series, pages 29–38, août 1995.

[Chi98] Changching Chiu. Merging Multiple Light Fields. Mémoire de maîtrise, University of British-Columbia, Computer Science Department, Imager lab., octobre 1998. À paraître.

[CLF98] Emilio Camahort, Apostolos Lerios et Donald Fussell. Uniformly Sampled Light Fields. Dans *Proceedings of Eurographics Workshop on Rendering '98*, édité par George Drettakis et Nelson Max, pages 117–130, juin 1998.

[COFHZ98] Daniel Cohen-Or, Gadi Fibich, Dan Halperin et Eyal Zadicario. Conservative Visibility and Strong Occlusion for Viewspace Partitioning of DEnsely Occluded Scenes. Dans *Computer Graphics Forum (EUROGRAPHICS '98 Proceedings)*, édité par M. Göbel et F. Nunes Ferreira, volume 17, pages C–243–C–253, septembre 1998.

[Coo86] Robert L. Cook. Stochastic Sampling in Computer Graphics. *ACM Transactions on Graphics*, 5(1) :51–72, janvier 1986.

[CT97] Satyan Coorg et Seth Teller. Real-Time Occlusion Culling for Models with Large Occluders. Dans *Proceedings of the Symposium on Interactive 3D Graphics*, pages 83–90, avril 1997.

[CW93a] Shenchang Eric Chen et Lance Williams. View Interpolation for Image Synthesis. Dans *Computer Graphics (SIGGRAPH '93 Proceedings)*, édité par James T. Kajiya, volume 27, pages 279–288, août 1993.

[CW93b] Michael F. Cohen et John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, 1993.

[DDP96] Frédo Durand, George Drettakis et Claude Puech. The 3D Visibility Complex : A New Approach to the Problems of Accurate Visibility. Dans *Proceedings of Eurographics Workshop on Rendering '96*, édité par Xavier Pueyo et Peter Schröder, pages 245–256, juin 1996.

[DDP97] Frédo Durand, George Drettakis et Claude Puech. The Visibility Skeletion : A Powerful and Efficient Multi-Purpose Global Visibility Tool. Dans *SIGGRAPH 97 Conference Proceedings*, édité par Turner Whitted, Annual Conference Series, pages 89–100, août 1997.

[Deb98] Paul E. Debevec. Rendering Synthetic Objects into Real Scenes : Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography. Dans *SIGGRAPH 98 Conference Proceedings*, édité par Michael Cohen, Annual Conference Series, pages 189–198, août 1998.

[DF97] Joel DeYoung et Alain Fournier. Properties of Tabulated Bidirectional Reflectance Distribution Functions. Dans *Proceedings of Graphics Interface '97*, édité par Wayne E. Davis, Marilyn Mantei et Victor Klassen, pages 47–55, mai 1997.

[DM97] Paul E. Debevec et Jitendra Malik. Recovering High Dynamic Range Radiance Maps from Photographs. Dans *SIGGRAPH 97 Conference Proceedings*, édité par Turner Whitted, Annual Conference Series, pages 369–378, août 1997.

[DRB97] George Drettakis, Luc Robert et Sylvain Bougnoux. Interactive Common Illumination for Computer Augmented Reality. Dans *Proceedings of Eurographics Workshop on Rendering '97*, édité par Julie Dorsey et Philipp Slusallek, pages 45–56, juin 1997.

[DTM96] Paul E. Debevec, Camillo J. Taylor et Jitendra Malik. Modeling and Rendering Architecture from Photographs : A Hybrid Geometry- and Image-Based Approach. Dans *SIGGRAPH 96 Conference Proceedings*, édité par Holly Rushmeier, Annual Conference Series, pages 11–20, août 1996.

[FA91] W. T. Freeman et Edward H. Adelson. The Design and Use of Steerable Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9) :891–906, septembre 1991.

[FGR93] Alain Fournier, Atjeng S. Gunawan et Chris Romanzin. Common Illumination Between Real and Computer Generated Scenes. Dans *Proceedings of Graphics Interface '93*, pages 254–262, mai 1993.

[FKN80] H. Fuchs, Z. M. Kedem et B. F. Naylor. On Visible Surface Generation by a Priori Tree Structures. Dans *Computer Graphics (SIGGRAPH '80 Proceedings)*, volume 14, pages 124–133, juillet 1980.

[Fou92] Alain Fournier. Normal Distribution Functions and Multiple Surfaces. Dans *Proceedings of Graphics Interface '92, Workshop on Local Illumination*, pages 45–52, mai 1992.

[Fou95] Alain Fournier. Illumination Problems in Computer Augmented Reality. Rapport technique, University of British Columbia, 1995.

[FvDFH90] James D. Foley, Andries van Dam, Steven K. Feiner et John F. Hughes. *Computer Graphics, Principles and Practice*. Addison-Wesley, deuxième edition, 1990.

[Ger87] Ron Gershon. *The Use of Color in Computational Vision*. Thèse de doctorat, Dept. of Computer Science, University of Toronto, 1987.

[GGC97] Xianfeng Gu, Steven J. Gortler et Michael F. Cohen. Polyhedral Geometry and the Two-Plane Parameterization. Dans *Proceedings of Eurographics Workshop on Rendering '97*, édité par Julie Dorsey et Philipp Slusallek, pages 1–12, juin 1997.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski et Michael F. Cohen. The Lumigraph. Dans *SIGGRAPH 96 Conference Proceedings*, édité par Holly Rushmeier, Annual Conference Series, pages 43–54, août 1996.

[GGSS89] Nader Gharachorloo, Satish Gupta, Robert F. Sproull et Ivan E. Sutherland. A Characterization of Ten Rasterization Techniques. Dans *Computer Graphics (SIGGRAPH '89 Proceedings)*, édité par Jeffrey Lane, volume 23, pages 355–368, juillet 1989.

[GK93] Ned Greene et M. Kass. Hierarchical Z-Buffer Visibility. Dans *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 231–240, 1993.

[Gla89] Édité par Andrew Glassner. *An Introduction to Ray Tracing*. Academic Press, 1989.

[Gla95] Andrew S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, 1995. Volumes I & II.

[GM90] Ziv Gigus et Jitendra Malik. Computing the Aspect Graph for Line Drawings of Polyhedral Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2) :113–122, février 1990.

[Gou71] Henri Gouraud. Continuous Shading of Curved Surfaces. *IEEE Transactions on Computers*, C-20(6) :623–629, juin 1971.

[Gre86] Ned Greene. Environment Mapping and Other Applications of World Projections. *IEEE Computer Graphics and Applications*, 6(11) :21–29, novembre 1986.

[GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg et Bennett Battaile. Modelling the Interaction of Light Between Diffuse Surfaces. Dans *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 212–22, juillet 1984.

[Gun91] Atjeng S. Gunawan. Estimating the Illuminant Color of a Scene from its Image Shading. Dans *Proceedings of the 1991 Western Computer Graphics Symposium*, pages 29–30, avril 1991.

[Hae92] P. Haeberli. Synthetic Lighting for Photography, janvier 1992. http ://www.sgi.com/grafica/synth/index.html.

[Hec86a] Paul S. Heckbert. Filtering by Repeated Integration. Dans *Computer Graphics (SIGGRAPH '86 Proceedings)*, édité par David C. Evans et Russell J. Athay, volume 20, pages 315–321, août 1986.

[Hec86b] Paul S. Heckbert. Survey of Texture Mapping. *IEEE Computer Graphics and Applications*, 6(11) :56–67, novembre 1986.

[Hec90] Paul S. Heckbert. Adaptive Radiosity Textures for Bidirectional Ray Tracing. Dans *Computer Graphics (SIGGRAPH '90 Proceedings)*, édité par Forest Baskett, volume 24, pages 145–154, août 1990.

[Hec92] Paul Heckbert. Discontinuity Meshing for Radiosity. Dans *Proceedings of Eurographics Workshop on Rendering '92*, pages 203–226, mai 1992.

[HKSS98] Wolfgang Heidrich, Jan Kautz, Philipp Slusallek et Hans-Peter Seidel. Canned Lightsources. Dans *Proceedings of Eurographics Workshop on Rendering '98*, édité par George Drettakis et Nelson Max, pages 293–300, juin 1998.

[Jen95] Henrik Wann Jensen. Importance Driven Path Tracing using the Photon Map. Dans *Proceedings of Eurographics Workshop on Rendering '95*, pages 326–335, juin 1995.

[Kaj86] James T. Kajiya. The Rendering Equation. Dans *Computer Graphics (SIGGRAPH '86 Proceedings)*, édité par David C. Evans et Rusell J. Athay, volume 20, pages 143–150, août 1986.

[KSK88] G. J. Klinker, S. A. Shafer et T. Kanade. The Measurement of Highlights in Color Images. *International Journal of Computer Vision*, 2 :7–32, 1988.

[Lal98] Paul Lalonde, mars 1998. Communication personnelle.

[LBS90] Hsien-Che Lee, Edwin J. Breneman et Carl P. Schulte. Modeling Light Reflection for Computer Color Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(4) :402–409, avril 1990.

[Lee88] Hsien-Che Lee. Estimating the Illuminant Color from the Shading of a Smooth Surface. Rapport technique AIM-1068, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, août 1988.

[Lev97] Mark Levoy. Expanding the Horizons of Image-Based Modeling and Rendering. SIGGRAPH 1997 Panel on Image-Based Rendering, août 1997.

[Lev98] Mark Levoy. The Digital Michelangelo Project. Stanford Computer Forum conference, mars 1998.

[LF94] Stéphane Laveau et Olivier Faugeras. 3-D Scene Representation as a Collection of Images and Fundamental Matrices. Rapport technique RR-2205, INRIA, février 1994.

[LF96] Robert R. Lewis et Alain Fournier. Light-Driven Global Illumination with a Wavelet Representation of Light Transport. Dans *Proceedings of Eurographics Workshop on Rendering '96*, édité par Xavier Pueyo et Peter Schröder, pages 11–20, juin 1996.

[LF97] Paul Lalonde et Alain Fournier. A Wavelet Representation of Reflectance Functions. *IEEE Transactions on Visualization and Computer Graphics*, 3(4) :329–336, octobre 1997.

[LH96] Marc Levoy et Pat Hanrahan. Light Field Rendering. Dans *SIGGRAPH 96 Conference Proceedings*, édité par Holly Rushmeier, Annual Conference Series, pages 31–42, août 1996.

[Lip80] A. Lippman. Movie-Maps : an Application of the Optical Videodisc to Computer Graphics. Dans *Computer Graphics (SIGGRAPH '80 Proceedings)*, volume 14, pages 32–42, juillet 1980.

[LS95] John Lansdown et Simon Schofield. Expressive Rendering : A Review of Nonphotorealistic Techniques. *IEEE Computer Graphics and Applications*, 15(3) :29–37, mai 1995.

[LTG92] Daniel Lischinski, Filippo Tampieri et Donald P. Greenberg. Discontinuity Meshing for Accurate Radiosity. *IEEE Computer Graphics and Applications*, 12(6) :25–39, novembre 1992.

[LW93] Eric P. Lafortune et Yves D. Willems. Bi-directional Path Tracing. Dans *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, édité par H. P. Santo, pages 145–153, décembre 1993.

[MB95] Leonard McMillan et Gary Bishop. Plenoptic Modeling : An Image-Based Rendering System. Dans *SIGGRAPH 95 Conference Proceedings*, édité par Robert Cook, Annual Conference Series, pages 39–46, août 1995.

[MMB97] William R. Mark, Leonard McMillan et Gary Bishop. Post-Rendering 3D Warping. Dans *Proceedings of the Symposium on Interactive 3D Graphics*, pages 7–16, avril 1997.

[MN88] Don P. Mitchell et Aru N. Netravali. Reconstruction Filters in Computer Graphics. Dans *Computer Graphics (SIGGRAPH '88 Proceedings)*, édité par John Dill, volume 22, pages 221–228, août 1988.

[NDW93] Jackie Neider, Tom Davis et Mason Woo. *OpenGL Programming Guide*. Addison-Wesley, 1993.

[Ney95a] Fabrice Neyret. Animated Texels. Dans *Computer Animation and Simulation '95*, édité par Dimitri Terzopoulos et Daniel Thalmann, pages 97–103, septembre 1995.

[Ney95b] Fabrice Neyret. A General and Multiscale Model for Volumetric Textures. Dans *Proceedings of Graphics Interface '95*, édité par Wayne A. Davis et Przemyslaw Prusinkiewicz, pages 83–91, mai 1995.

[NINT89] Eihachiro Nakamae, Takao Ishizaki, Tomoyuki Nishita et Shinichi Takita. Compositing 3D Images with Antialiasing and Various Shading Effects. *IEEE Computer Graphics and Applications*, 9(2) :21–29, mars 1989.

[NRH+77] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg et T. Limperis. Geometric Considerations and Nomenclature for Reflectance. Monograph 161, National Bureau of Standards (US), octobre 1977.

[NSD94] Jeffry S. Nimeroff, Eero Simoncelli et Julie Dorsey. Efficient Re-rendering of Naturally Illuminated Environments. Dans *Proceedings of Eurographics Workshop on Rendering '94*, pages 359–373, juin 1994.

[OB98] Manuel M. de Oliveira Neto et Gary Bishop. Dynamic Shading in Image-Based Rendering. Rapport technique TR98-023, University or North Carolina at Chapel Hill, mai 1998.

[OLF98] Office de la Langue Française du Québec. Vocabulaire d'Internet Plus. Site web, 1998. URL :http ://www.olf.gouv.qc.ca/neuf/pages/vocinter2/internet_C.html.

[PCD+97] Kari Pulli, Michael Cohen, Tom Duchamp, Hugues Hoppe, Linda Shapiro et Werner Stuetzle. View-based Rendering : Visualizing Real Objects from Scanned Range and Color Data. Dans *Proceedings of Eurographics Workshop on Rendering '97*, édité par Julie Dorsey et Philipp Slusallek, pages 23–34, juin 1997.

[PD84] Thomas Porter et Tom Duff. Compositing Digital Images. Dans *Computer Graphics (SIGGRAPH '84 Proceedings)*, édité par Hank Christiansen, volume 18, pages 253–259, juillet 1984.

[PF95] Pierre Poulin et Alain Fournier. Painting Surface Characteristics. Dans *Proceedings of Eurographics Workshop on Rendering '95*, pages 119–129, juin 1995.

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

[PRJ97] Pierre Poulin, Karim Ratib et Marco Jacques. Sketching Shadows and Highlights to Position Lights. Dans *Proceedings of Computer Graphics International 97*, pages 56–63, juin 1997.

[RB98] Paul Rademacher et Gary Bishop. Multiple Center-of-Projection Images. Dans *SIGGRAPH 98 Conference Proceedings*, édité par Michael Cohen, Annual Conference Series, pages 199–206, août 1998.

[RBMT98] Holly Rushmeier, Fausto Bernardini, Joshua Mittleman et Gabriel Taubin. Acquiring Input for Rendering at Appropriate Levels of Detail : Digitizing a Pietà. Dans *Proceedings of Eurographics Workshop on Rendering '98*, pages 81–92, juin 1998.

[Rob65] L. Roberts. Machine Perception of Three-Dimensional Solids. Dans *Optical and Electro-Optical Information Processing*, édité par J. Tippett et al., pages 159–197. MIT Press, 1965.

[Rom95] Christopher A. Romanzin. Aspects of Image Reshading. Mémoire de maîtrise, University of British-Columbia, Computer Science Department, Imager laboratory, 1995.

[Rus98] Szymon Rusinkiewicz. A New Change of Variables for Efficient BRDF Representation. Dans *Proceedings of Eurographics Workshop on Rendering '98*, édité par George Drettakis et Nelson Max, pages 11–22, juin 1998.

[SAWG91] Francois X. Sillion, James R. Arvo, Stephen H. Westin et Donald P. Greenberg. A Global Illumination Solution for General Reflectance Distributions. Dans *Computer Graphics (SIGGRAPH '91 Proceedings)*, édité par Thomas W. Sederberg, volume 25, pages 187–196, juillet 1991.

[SB96] Alvy Ray Smith et James F. Blinn. Blue Screen Matting. Dans *SIGGRAPH 96 Conference Proceedings*, édité par Holly Rushmeier, Annual Conference Series, pages 259–268, août 1996.

[SCG97] Peter-Pike Sloan, Michael F. Cohen et Steven J. Gortler. Time Critical Lumigraph Rendering. Dans *Proceedings of the Symposium on Interactive 3D Graphics*, pages 17–24, avril 1997.

[SD95] Steven M. Seitz et Charles R. Dyer. Physically-Valid View Synthesis by Image Interpolation. Dans *Proceedings of Workshop on Representation of Visual Scenes '95*, pages 18–25, juin 1995.

[SD96] Steven M. Seitz et Charles R. Dyer. View Morphing : Synthesizing 3D Metamorphoses Using Image Transforms. Dans *SIGGRAPH 96 Conference Proceedings*, édité par Holly Rushmeier, Annual Conference Series, pages 21–30, août 1996.

[SGHS98] Jonathan Shade, Steven Gortler, Li-Wei He et Richard Szeliski. Layered Depth Images. Dans *SIGGRAPH 98 Conference Proceedings*, édité par Michael Cohen, Annual Conference Series, pages 231–242, août 1998.

[SHW92] Steve Shafer, Glen Healey et Larry Wolff. *Physics Based Vision Principles and Practice : Color*. A K Peters, 1992.

[SKv+92] Mark Segal, Carl Korobkin, Rolf van Widenfelt, Jim Foran et Paul E. Haeberli. Fast Shadows and Lighting Effects using Texture Mapping. Dans *Computer*

Graphics (SIGGRAPH '92 Proceedings), édité par Edwin E. Catmull, volume 26, pages 249–252, juillet 1992.

[Som59] D. M. Y. Sommerville. *Analytical Geometry of Three Dimensions*. Cambridge University Press, 1959.

[SS97] Richard Szeliski et Heung-Yeung Shum. Creating Full View Panoramic Image Mosaics and Environment Maps. Dans *SIGGRAPH 97 Conference Proceedings*, édité par Turner Whitted, Annual Conference Series, pages 251–258, août 1997.

[SSS74] Ivan E. Sutherland, Robert F. Sproull et Robert A. Schumacker. A Characterization of Ten Hidden-Surface Algorithms. *ACM Computing Surveys*, 6(1) :1–55, mars 1974.

[Ste95] N. F. Stewart. Science and Computer Science. *ACM Computing Surveys*, 27(1) :39–41, mars 1995.

[SWI97] Yoichi Sato, Mark D. Wheeler et Katsushi Ikeuchi. Object Shape and Reflectance Modeling from Observation. Dans *SIGGRAPH 97 Conference Proceedings*, édité par Turner Whitted, Annual Conference Series, pages 379–388, août 1997.

[Sze93] Richard Szeliski. Rapid Octree Construction from Image Sequences. *Computer Vision, Graphics, and Image Processing. Image Understanding*, 58(1) :23–32, juillet 1993.

[Tau95] Gabriel Taubin. A Signal Processing Approach to Fair Surface Design. Dans *SIGGRAPH 95 Conference Proceedings*, édité par Robert Cook, Annual Conference Series, pages 351–358, août 1995.

[Tel92] Seth J. Teller. Computing the Antipenumbra of an Area Light Source. Dans *Computer Graphics (SIGGRAPH '92 Proceedings)*, édité par Edwin E. Catmull, volume 26, pages 139–148, juillet 1992.

[TH93] Seth Teller et Pat Hanrahan. Global Visibility Algorithms for Illumination Computations. Dans *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 239–246, 1993.

[TL94] Greg Turk et Marc Levoy. Zippered Polygon Meshes from Range Images. Dans *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, édité par Andrew Glassner, Computer Graphics Proceedings, Annual Conference Series, pages 311–318, juillet 1994.

[TT98] Troll Tech A/S. The Qt toolkit, version 1.40, 1998. URL : http ://www.troll.no.

[WA77] K. Weiler et K. Atherton. Hidden Surface Removal using Polygon Area Sorting. *Computer Graphics (SIGGRAPH '77 Proceedings)*, 11(2) :214–222, juillet 1977.

[War69] J. Warnock. A Hidden-Surface Algorithm for Computer Generated Half-Tone Pictures. Rapport technique TR 4–15, NTIS AD-733 671, University of Utah, Computer Science Department, 1969.

[WHG84] Hank Weghorst, Gary Hooper et Donald P. Greenberg. Improved Computational Methods for Ray Tracing. *ACM Transactions on Graphics*, 3(1) :52–69, janvier 1984.

[Whi80] Turner Whitted. An Improved Illumination Model for Shaded Display. *Communications of the ACM*, 23(6) :343–349, 1980.

[WHON97] Tien-Tsin Wong, Pheng-Ann Heng, Siu-Hang Or et Wai-Yin Ng. Image-based Rendering with Controllable Illumination. Dans *Proceedings of Eurographics Workshop on Rendering '97*, édité par Julie Dorsey et Philipp Slusallek, pages 13–22, juin 1997.

[Wil78] Lance Williams. Casting Curved Shadows on Curved Surfaces. Dans *Computer Graphics (SIGGRAPH '78 Proceedings)*, volume 12, pages 270–274, août 1978.

[Wil83] Lance Williams. Pyramidal Parametrics. Dans *Computer Graphics (SIGGRAPH '83 Proceedings)*, volume 17, pages 1–11, juillet 1983.

[WL97a] Gregory Ward Larson. LogLuv encoding for TIFF images, 1997. Silicon Graphics. URL : http ://www.sgi.com/Technology/pixformat/tiffluv.html.

[WL97b] Gregory Ward Larson. A Proposal to Develop a High Dynamic Range Pixel Encoding, 1997. Silicon Graphics. URL : http ://www.sgi.com/Technology/pixformat/proposal.html.

[WLRP97] Gregory Ward Larson, Holly Rushmeier et Christine Piatko. A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes. Rapport technique LBNL 39882, Lawrence Berkeley National Laboratory, 1997.

[Wol90] George Wolberg. *Digital Image Warping*. IEEE Computer Society Press, 1990.

[XX98] Xing Xing. Yuan Ming Yuan project (Garden of Perfect Brightness), 1998. URL : http ://www.xing-xing.com.

[YM98] Yizhou Yu et Jitendra Malik. Recovering Photometric Properties Of Architectural Scenes From Protographs. Dans *SIGGRAPH 98 Conference Proceedings*, édité par Michael Cohen, Annual Conference Series, pages 207–217, août 1998.

[ZMHI97] Hansong Zhang, Dinesh Manocha, Thomas Hudson et Kenneth E. Hoff III. Visibility Culling Using Hierarchical Occlusion Maps. Dans *SIGGRAPH 97 Conference Proceedings*, édité par Turner Whitted, Annual Conference Series, pages 77–88, août 1997.

Annexe A

Glossaire

Cette annexe contient quelques traductions de l'anglais pour certains termes utilisés dans le présent document. Nous avons tenté de restreindre le nombre de termes de l'anglais utilisés directement dans le texte. Ce faisant, il a fallu traduire plusieurs termes qui n'ont pas encore d'équivalents officiels dans la langue française ou encore qui ne sont pas d'utilisation courante et auquels le lecteur pourrait ne pas être familier. Ces traductions sont regroupées ici.

apposition de textures $\longleftrightarrow$ *texture mapping*

balayage de lignes $\longleftrightarrow$ *scan conversion* ou *rasterization*

bloqueur $\longleftrightarrow$ *occluder*

carte d'environnement $\longleftrightarrow$ *environment map*

carte d'illumination $\longleftrightarrow$ *illumination map*

carte d'ombres $\longleftrightarrow$ *shadow map*

carte de disparité $\longleftrightarrow$ *disparity map*

correspondance stéréo $\longleftrightarrow$ *stereo matching*

dessin de lignes $\longleftrightarrow$ *line drawing*

déformation d'image $\longleftrightarrow$ *image morphing*

échantillonnage ponctuel $\longleftrightarrow$ *point sampling*

éventail de fréquences $\longleftrightarrow$ *dynamic range*

événements de visibilité ⟷ *visibility event*

facteur de forme ⟷ *form factor*

fusion alpha ⟷ *alpha blending*

graphe d'aspect ⟷ *aspect graph*

illumination globale ⟷ *global illumination*

illumination locale ⟷ *local illumination*

image d'occlusion hiérarchique ⟷ *hierarchical occlusion map*

image de profondeur ⟷ *depth map*

image de radiance ⟷ *radiance map*

image panoramique ⟷ *panorama, panoramic image*

irradiance ⟷ *irradiance*

lancer de rayons (tracer de rayons sans réflexions ni réfractions) ⟷ *ray casting*

ligne de balayage ⟷ *scanline*

liste d'affichage ⟷ *display list*

logiciel de rendu, moteur de rendu ⟷ *renderer, rendering engine*

modèle d'illumination ⟷ *illumination model*

numériseur de profondeur ⟷ *range scanner*

paramétrisation à deux plans, 2PP ⟷ *two-plane parameterization*

paramétrisation à deux sphères, 2SP ⟷ *two-sphere parameterization*

paramétrisation sphère-plan, SPP ⟷ *sphere-plane parameterization*

pipeline graphique ⟷ *graphics pipeline*

radiance ⟷ *radiance*

radiosité (par éléments finis) ⟷ *radiosity*

reflets spéculaires ⟷ *highlights*

rendu basé sur des images ⟷ *image-based rendering*

rendu différentiel ⟷ *differential rendering*

rendu incrémental ⟷ *incremental rendering*

reprojection d'image ⟷ *forward image mapping*

squelette de visibilité $\longleftrightarrow$ *visibility skeleton*

tampon d'objet $\longleftrightarrow$ *item buffer*

tampon de profondeur $\longleftrightarrow$ *z-buffer*

tampon de profondeur hiérarchique $\longleftrightarrow$ *hierarchical z-buffer*

tampon image secondaire $\longleftrightarrow$ *back-buffer*

texel volumétrique $\longleftrightarrow$ *volumetric texel*

texture de relief $\longleftrightarrow$ *bump map*

tracer de chemins $\longleftrightarrow$ *path tracing*

tracer de rayons $\longleftrightarrow$ *ray tracing*

volume englobant ou boîte englobante $\longleftrightarrow$ *bounding box, bounding volume*

Index