UNIVERSITE D'AUVERGNE
CLERMONT-FD1

LAIC
LABORATOIRE D'ALGORITHME ET IMAGE DE CLERMONT-FERRAND

*N° attribué par la bibliothèque*

**Université d'Auvergne**
**École doctorale Sciences pour l'Ingénieur**
**Laboratoire LAIC**

Thèse

présentée par

Pierre Y. Chatelier

et soutenue

**le 4 décembre 2006**

en vue de l'obtention du

# Doctorat de l'Université d'Auvergne

**Spécialité :** Informatique

## Titre :

# Une approche de la radiosité par voxels, application à la synthèse d'images

---

**Directeur de thèse :** Rémy Malgouyres

---

**Jury**

| | |
|---|---|
| M. Éric Andres | Rapporteur |
| M. Didier Arquès | Rapporteur |
| M. Bernard Péroche | Rapporteur |
| M. Fabien Feschet | Examinateur |
| M. Rémy Malgouyres | Directeur de thèse |

UNIVERSITE D'AUVERGNE
CLERMONT-FD1

LAIC
LABORATOIRE D'ALGORITHME ET IMAGE DE CLERMONT-FERRAND

*N° attribué par la bibliothèque*

**Université d'Auvergne**
**École doctorale Sciences pour l'Ingénieur**
**Laboratoire LAIC**

Thèse

présentée par

Pierre Y. Chatelier

et soutenue

**le 4 décembre 2006**

en vue de l'obtention du

# Doctorat de l'Université d'Auvergne

**Spécialité :** Informatique

## Titre :

# Une approche de la radiosité par voxels, application à la synthèse d'images

---

**Directeur de thèse :** Rémy Malgouyres

---

**Jury**

| | |
|---|---|
| M. Éric Andres | Rapporteur |
| M. Didier Arquès | Rapporteur |
| M. Bernard Péroche | Rapporteur |
| M. Fabien Feschet | Examinateur |
| M. Rémy Malgouyres | Directeur de thèse |

Cette thèse a été effectuée au sein du LAIC, Laboratoire d'Algorithme et Image de Clermont-Ferrand, équipe d'accueil 2146.
L'adresse du laboratoire est la suivante :

LAIC, IUT departement Informatique
B.P.86
63172 AUBIERE cedex
FRANCE

# Remerciements

Je ne connais pas de doctorant qui n'admette avoir souffert pendant sa thèse, pour les inquiétudes, le stress, et les nombreuses incertitudes qui lui sont associées. Bien entendu, je n'ai pas échappé à cela, même si j'ai connu des témoignages de situations encore bien moins enviables. Il s'agit d'une expérience douloureuse, qui apparaît cependant moins vive une fois qu'elle est passée, comme la plupart les élancements dont on a tout oublié sauf le soulagement qui s'ensuit.

Ce n'est cependant pas de désastrologie dont je veux faire état. Les gens qui m'ont entouré m'ont permis d'aller jusqu'au bout, et c'est à eux qu'il me brûle d'adresser des remerciements. À Rémy tout d'abord, mon principal soutien dans l'aventure, jamais avare du remonte-moral et de la ténacité qui ont pu me faire défaut. À Fabien évidemment, avec qui j'ai partagé bien plus qu'un bureau et dont les nombreuses connaissances ont aidé le thésard autant que l'informaticien. À Yan également, qui sait rompre la morosité quand elle s'insinue trop loin. À Jean-Pierre, Malika, Alex pour leur présence, leurs conseils, ou leurs remarques avisées.

À mes compagnons de chébèque, je ne sais quel merci donner. En ayant le bon goût de partager l'infortune, vous n'en êtes pas moins des amis fidèles. Alain, François, Furby, Guillaume, Jérôme, Pascal, Sophie, vous avez été là ; et c'est aussi cette présence qui vous rend si précieux.

Aux suivants, Rita, Thibault, Fabien, Alexandre et Mustafa, je ne peux que vous remercier à la hauteur du courage que je vous souhaite. Puissiez-vous tout au moins ne pas en avoir trop besoin !

Je ne saurais manquer non plus de remercier François Delobel, dont le parrainage pour l'enseignement m'a été la plus belle entrée en matière dont j'aie pu rêver. Et pour ajouter encore un François, c'est à François Gaillard que j'adresse un merci mêlé de considération et de respect. Je ne puis que m'étonner de la richesse de ses enseignements, et de ses discussions intempestives, dont j'espère avoir su profiter à la hauteur de leur diversité.

C'est enfin à mes proches, famille et presque-famille, que j'adresse des pensées reconnaissantes. Pour se restreindre à quelques mots : la vie est riche à vos côtés.

# Introduction

L'image de synthèse est un domaine de recherche en constante évolution, car la puissance grandissante des machines remet toujours en question le compromis entre réalisme et coût des calculs nécessaire à l'obtenir. Des algorithmes trop coûteux deviennent abordables et leur utilisation plus courante. De nouvelles recherches peuvent alors être consacrées à des domaines peu usités auparavant.

L'équation de radiosité est une équation physique dont la solution représente un éclairage réaliste d'une scène donnée. Son implantation dans l'imagerie de synthèse est l'exemple d'un algorithme fondamentalement coûteux, mais dont les grandes lignes ont pu être tracées depuis plusieurs années. Les machines courantes sont maintenant capables de produire des images utilisant la radiosité en quelques minutes, ce qui rend cet algorithme beaucoup plus accessible et permet de déployer à plus grande échelle les méthodes préalablement mises au point.

Ma thèse a été l'occasion d'expérimenter une façon inédite d'aborder le problème. Il s'agit d'introduire l'utilisation de notions mathématiques inhabituelles pour des calculs intermédiaires, qui permettent alors de réordonner l'ensemble des calculs de la méthode. Plus précisément, il s'agit d'utiliser des surfaces discrètes voxelisées plutôt que des maillages, ce qui transforme complètement le problème de la visibilité des éléments entre eux dans la scène. Si certaines idées peuvent être reprises dans les algorithmes classiques, il faut pratiquement repenser l'intégralité du calcul. Ma thèse montre les différents points abordés pour concevoir cette nouvelle méthode.

Comparée aux algorithmes classiques, elle permet notamment d'utiliser une discrétisation plus fine de la scène, grâce à une complexité faible et un temps de calcul stable et prévisible. Il n'y a pas d'alourdissement des calculs au fur et à mesure des itérations, et peu d'itérations sont nécessaires par le biais de l'approche très « systématique » des opérations.

L'organisation du présent manuscrit ne reflète pas directement la chronologie du déroulement de ma thèse. Cette chronologie me paraît cependant intéressante pour expliquer les thèmes de recherche qui ont été abordés.

Bien évidemment, j'ai d'abord commencé par me documenter sur la radiosité et les techniques habituelles, pour comprendre les problèmes et les façons d'y remédier. La radiosité fait appel à de nombreuses notions d'imagerie de synthèse. On y trouve les problématiques de représentation de la scène, de l'éclairage, des couleurs, ainsi que les techniques de lancer de rayon, de projection, d'échantillonnage, et la prise en compte de phénomènes physiques d'optique géométrique ou ondulatoire. La multiplicité des difficultés implique une grande diversité algorithmique, assez longue à appréhender dans son ensemble. Il s'agit des chapitres 1 et 2. Le chapitre 1 traite des différentes problématiques de l'imagerie de synthèse, tandis que dans le chapitre 2 ont été déportés, pour être détaillés, les algorithmes de radiosité et

d'illumination globale, qui sont liés.

Le sujet de la thèse était cependant déjà défini, à savoir la conception d'un algorithme de radiosité utilisant des données inhabituelles : des surfaces discrétisées en voxels (petits cubes). Un premier algorithme avait déjà été mis au point, qu'il s'agissait d'améliorer. Ce premier algorithme faisant déjà usage de la géométrie discrète, j'ai également dû me plonger dans ce domaine mathématique pour prendre connaissance des algorithmes connus et utilisables. C'est dans le chapitre 3 que j'expose les différentes notions qui ont pu m'être utiles.

La connaissance des algorithmes de droites 3D discrètes m'a effectivement permis de mettre au point une version très améliorée de l'algorithme de radiosité initial, en produisant une version quasi-optimale de la résolution. La conception de cet algorithme est détaillée dans le chapitre 4.

Cette première étape étant franchie, je me suis interrogé sur le champ d'action de cet algorithme. Plutôt que de me cantonner à la radiosité des images de synthèse, et à cause des limitations mises en lumières dans la section du même nom, j'ai cherché à appliquer le code à un autre domaine de la physique : le *transfert radiatif* dans les nuages. Cette tentative n'a pas été entièrement couronnée de succès, même si les premiers résultats sont encourageants. Ceux-ci sont détaillés dans le chapitre 6, qui traite des extensions de mon algorithme.

En revanche, la mise au point des *fonctions de phases* nécessaires à ces calculs de transfert radiatif m'a donné à réfléchir sur la représentation peu coûteuse de fonctions sphériques 2D. Je suis retombé là sur une problématique classique de l'éclairage en image de synthèse : les fonctions *BRDF* des matériaux. Dans le contexte des nuages, j'ai pu produire une nouvelle méthode d'encodage de ces fonctions. Étrangement, cette méthode est simple et il est étonnant qu'elle n'ait pas été déjà expérimentée. Nous n'avons pas trouvé de références antérieures à une telle méthode. La conception de ces BRDFs fait l'objet du chapitre 5. Grâce à son application directe au calcul de l'illumination globale qui est un cas plus général de la radiosité, il semble plus logique de le positionner à la suite du chapitre 4.

Le chapitre 6 présente quant à lui d'autres extensions de mon algorithme qui n'ont pas encore pu être pleinement expérimentées. Il s'agit de raffinements, de méthodes mixtes et surtout de parallélisation ; soit des perspectives d'évolution sur une base algorithmique solide. C'est également là que se trouvent les explications sur le transfert radiatif dans les nuages. Nous présentons aussi le projet logiciel qui doit permettre de comparer l'algorithme de radiosité aux algorithmes existants, en l'implantant dans un contexte facilitant les tests.

# Table des matières

# Table des figures

# Glossaire

**BBQ** : *Background Boundary Quantization*. Discrétisation privilégiant l'extérieur de l'objet.
**BRDF** : *Bidirectional Reflectance Distribution Fonction*. Cette fonction à deux paramètres (direction d'entrée, direction de sortie) renvoie le rapport entre énergie incidente et énergie sortante à un point d'impact.
**B-Splines** : type de courbe paramétrée utilisant une base de fonctions particulièrement adaptée à la modélisation de courbes ou surfaces déformables localement.
**CSG** : *Constructive Solid Geometry*. La géométrie de construction de solides procède à des opérations ensemblistes pour générer des solides complexes à partir de primitives simples.
**Facteur de forme** : mesure géométrique caractérisant notamment l'angle solide entre deux éléments d'une scène.
**Fonction de phase** : Distribution caractérisant la probabilité de changement de direction d'un photon.
**GIQ** : *Grid Intersection Quantization*. Discrétisation privilégiant la proximité aux noeuds de la grille.
**Hémicube** : demi-cube dont les cellules sur chaque face servent d'écran virtuel de projection 3D $\rightarrow$ 2D.
**Mesh** : Maillage.
**MIMD** : *Multiple Instruction Multiple Data*. Application simultanée de différentes instructions à différentes données.
**MPI** : *Message passing Interface*. Bibliothèque logicielle pour gérer la communication entre machines, notamment dans les programmes parallèles.
**NURBS** : *Non Uniform Rational B-Splines*. Type de courbe paramétrée utilisant un ratio de B-Splines.
**OBQ** : *Object Boundary Quantization*. Discrétisation privilégiant l'intérieur de l'objet.
**Octree** : Structure de données hiérarchique sous forme d'arbre d'arité 8, utilisée notamment pour représenter des volumes.
**Patch** : morceau de surface constituant un élément de maillage.
**Rasterization** : Tramage.
**SIMD** : *Single Instruction Multiple Data*. Application simultanée de la même instruction à différentes données.
**Spel** : élément d'un espace discret
**Thread** : Fil d'exécution d'un programme. Un processus peut disposer de plusieurs fils d'exécution concurrents dans le même espace mémoire.
**Voxel** : cube élémentaire, c'est-à-dire spel de dimension 3.

# Première partie

# Domaines de travail

# Chapitre 1

# Synthèse d'images

La *Synthèse d'images* est la génération d'images par ordinateur. Que cela soit réalisé dans un but esthétique, comme dans les applications multimedia, ou utilitaire, comme l'architecture, la représentation *réaliste* d'une scène se heurte aux mêmes problèmes. La difficulté réside dans le compromis à trouver entre la « qualité de la représentation » et la charge de calculs sous-jacents. C'est un *moteur de rendu* qui est chargé de produire une image, ou une animation (une suite d'images), à partir des données choisies pour représenter une scène.

La qualité de la représentation, terme volontairement flou, est difficile à préciser hors contexte. En effet, comme tout problème de modélisation, seul le positionnement du modèle par rapport au système réel donne un sens à ses caractéristiques, sous forme de qualités ou de défauts. On peut toutefois citer quelques exemples courants de critères permettant de juger de la qualité d'une image ou d'une animation générée par ordinateur :

- finesse de représentation des détails géométriques ;
- couleurs des objets (au sens d'aspect) ;
- texture des objets (au sens de micro-géométrie) ;
- comportement physique cohérent des objets ;
- représentation de phénomènes non géométriques (milieu absorbant, nuages de particules...) ;

Les domaines d'application de la synthèse d'images sont divers et ne conduisent pas aux même compromis. Un monde virtuel créé pour faire se déplacer des avatars a généralement une vocation de fidélité et d'esthétisme en temps réel, tandis que la visualisation de données privilégie la précision. *La réalité augmentée* est un cas particulier qui mélange les acquisitions d'un monde réel avec les productions d'un moteur de rendu. Dans tous les cas, l'immersion de l'utilisateur est d'autant plus réussie qu'il est difficile de distinguer ce qui est réel de ce qui ne l'est pas.

Le compromis guide ainsi les modèles sous-jacents, choisis en fonction de leur capacité à représenter certains phénomènes, que l'on peut négliger

ou non. Dans le cas de l'architecture, on peut par exemple exiger que la résistance des matériaux soit prise en compte, les forces exercées étant représentées par des couleurs, pour valider la structure du bâtiment et exhiber d'éventuelles faiblesses. On peut aussi demander que l'éclairage soit reproduit au mieux, pour aider l'architecte à agencer les pièces en les dotant des fonctions adaptées.
Dans le domaine de la *simulation*, un modèle peut être utilisé pour valider un autre modèle. En synthèse d'images, on peut par exemple tester un modèle de transport d'énergie en s'en servant comme composante *éclairement* d'un moteur de rendu (cf. section 1.2 page 25). Dans ce cas, les images générées doivent permettre de décider si le modèle de transport d'énergie est utilisable pour simuler le comportement de la lumière, dans le domaine qu'on a choisi de représenter.

Le présent chapitre introduit les différentes notions utilisées classiquement pour la synthèse d'images. Il traite de la modélisation géométrique (section 1.1 page suivante), des modèles d'éclairement (section 1.2 page 25), des problèmes de traitement (section 1.3 page 30) et des techniques de rendu (section 1.4 page 33).

## 1.1 Modélisation géométrique

La représentation informatique d'une scène tri-dimensionnelle repose en partie sur la modélisation géométrique des objets. Cette première étape nécessite déjà de fixer les limites du modèle ; en effet, en fonction de l'échelle du monde à représenter, il importe de définir ce qui est un objet représentable et ce qui ne l'est pas. Un monde virtuel est constitué le plus souvent d'objets macroscopiques ; dans ce cas, les « objets » microscopiques doivent être pris en compte par un modèle non géométrique si on ne veut pas négliger leur impact à l'échelle macroscopique. Par exemple, les particules d'un nuage de fumée s'élevant d'un cendrier représenteraient une quantité de données impossible à traiter en un temps raisonnable ; l'absorption locale de la lumière peut alors être déportée dans le modèle d'éclairement. Comme autre exemple, l'aspect granuleux d'un crépi va sans doute être déporté dans une texture (cf. section 1.1.7 page 24) pour renseigner le modèle d'éclairement. Si l'on désire visualiser des molécules, on peut décider d'une représentation géométrique des atomes, par exemple sous forme de sphères ; mais un objet macroscopique risque d'être hors de portée de calcul sous cette forme. À l'inverse, si l'on veut visualiser des galaxies, même une lune peut être trop petite pour être modélisée par un objet particulier.

Un modèle est donc généralement adapté à une échelle particulière. Si les données à traiter sont multi-échelle, il faut envisager d'utiliser un modèle multi-échelle [BM02, Guo95, Wes94], ou être capable d'assurer une transition entre des modèles différents.

### 1.1.1 Surfaces et volumes

Parmi les critères à considérer pour choisir une représentation, la question se pose de savoir si l'on peut représenter un objet par sa *surface* ou par son *volume*. En fonction du but recherché, l'une ou l'autre représentation peut être mieux adaptée. Les opérations booléennes de géométrie de construction de solides (cf. section 1.1.2) sont plus aisées avec des volumes, mais un objet complexe est souvent plus facile à représenter par des morceaux de surface.

Ce choix est également important dans les modèles d'éclairement (cf. section 1.2 page 25) puisque l'interaction de la lumière avec un objet peut être réduite aux interfaces (réflexion/réfraction simples d'après la loi de Snell-Descartes) ou nécessiter une analyse du trajet intérieur au volume (absorption, dispersion...)

### 1.1.2 Primitives et géométrie de construction de solides

En se laissant guider par les mathématiques, la démarche la plus intuitive pour représenter un objet complexe est de le subdiviser en un assemblage

d'objets plus simples et atomiques appelés *primitives*. En s'autorisant les opérations d'union, d'intersection et de différence, qui constitue la *géométrie de construction de solides*, ou *Constructive Solid Geometry* (CSG), de nombreuses formes peuvent ainsi être réalisées (Figure 1.1).

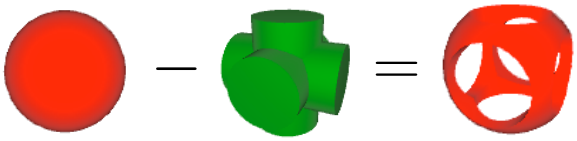


FIG. 1.1 – Exemple de *Constructive Solid Geometry* : soustraction de formes

En subdivisant assez finement un objet complexe, il est quasiment toujours possible de le représenter comme un assemblage de primitives. Cela dépend de la richesse et de la diversité de ces dernières. En revanche, le coût de représentation peut devenir prohibitif. La notion de primitive est donc étendue à des formes très modulables comme les maillages (cf. section 1.1.5 page suivante).

### 1.1.3 Surfaces implicites

On peut choisir, comme primitives de base, des objets représentables mathématiquement par une équation implicite : plan, sphère, ellipsoïde, tore, cylindre, parallélépipède rectangle...

Un des avantages des surfaces implicites est qu'elles sont paramétrables. En revanche, des perturbations locales sont plus difficiles à représenter et peuvent nécessiter de combiner plusieurs équations. Cela rend assez difficile l'interaction avec l'utilisateur. De plus, s'il est relativement aisé de manipuler mathématiquement des surfaces implicites, il n'est pas trivial de les représenter à l'écran. Le moteur de rendu doit en effet passer d'un modèle mathématique pur à un domaine d'image, ce qui est un problème difficile [BW97, SK01], notamment résolu par lancer de rayons (cf. section 1.4.4 page 34).

### 1.1.4 Surfaces paramétrées

En synthèse d'images, une surface paramétrée est une application de $\mathbb{R}^2$ dans $\mathbb{R}^3$, le domaine de départ étant souvent homéomorphe à un disque, et la surface générée continue. Une surface paramétrée est un modèle plus limitatif qu'une surface implicite, mais plus facile à manipuler. De plus, sa nature d'*application* est bien adaptée à des algorithmes de visualisation.

$$S(t,u) : \left| \begin{array}{ccc} \mathbb{R}^2 & \longrightarrow & \mathbb{R}^3 \\ (t,u) & \longmapsto & (t,u,t^2+u^2) \end{array} \right.$$

$$S(t,u) : \left| \begin{array}{ccc} \mathbb{R}^2 & \longrightarrow & \mathbb{R}^3 \\ (t,u) & \longmapsto & (t,u,t^2+u^2) \end{array} \right.$$

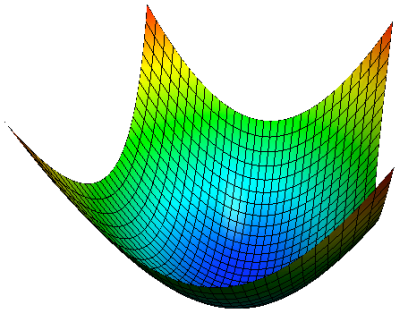


FIG. 1.2 – Une surface paramétrée paraboloïde. $x$, $y$ et $z$ sont des fonctions de $t$ et $u$.

### 1.1.5 Maillages

#### 1.1.5.1 Structure d'un maillage

Pour représenter un objet complexe, comme peut l'être un visage, il est assez satisfaisant de représenter sa surface comme un maillage. Un maillage est une discrétisation de la surface, en un nombre fini d'éléments. Un ensemble de triangles est une des représentations les plus simples d'un maillage. Mais à moins d'utiliser de petits triangles, l'effet produit par un tel maillage se rapproche d'une sculpture grossière, les facettes étant visibles. Ce problème peut être en partie résolu par le modèle d'éclairement, comme avec le lissage de Gouraud (cf. section 1.3.1.1 page 30. De façon géométrique, on peut aussi choisir d'utiliser des surfaces de Bézier, des B-Splines ou des NURBS[1] plutôt que des triangles. On apporte ainsi une information de courbure et des raccords lisses entre les éléments du maillage. Les surfaces de Bézier, les B-Splines et les NURBS sont des modèles de plus en plus complexes basés sur un calcul polynomial paramétré [FG99, Mal05].

#### 1.1.5.2 Optimisation du maillage

Le choix des points de contrôle du maillage fait partie de la définition de ce dernier, mais le choix des facettes est tout aussi déterminant. De « mauvaises » facettes conduisent à un résultat difficile à rendre agréable à l'œil. La constitution d'un bon maillage est une thématique de recherche. Les critères de qualité peuvent être la régularité, la forme, la taille, le nombre de ses éléments [BE92]. La triangulation de Delaunay est un outil de choix pour réaliser un tel maillage [Che87, KBT03].

La régularité du maillage ne signifie pas non plus qu'il est homogène sur toute la surface à représenter. Un objet complexe dispose généralement de zones « simples » nécessitant peu de détails, et de zones « perturbées »

[1] *Non-Uniform Rational B-Splines*

demandant une précision accrue. C'est à la fois le jeu des points de contrôle et des éventuels paramètres des éléments du maillage qui permettent de contrôler la complexité locale d'un objet.

### 1.1.6 Décomposition sur une base de fonctions

Certaines formes sont adaptées à une représentation sur une base de fonctions. Les ondelettes constituent un outil puissant de décomposition qui peut être appliqué à la représentation de formes géométriques [SS95]. Une telle technique est cependant plus utile dans le stockage de l'information que dans sa manipulation. Il est plutôt envisageable, pour un objet stocké sous cette forme, de produire un échantillonnage de sa surface et d'en déduire une approximation sous forme de maillage, utilisable pour les calculs d'un moteur de rendu.

### 1.1.7 Textures et Couleur

Physiquement, la couleur d'un objet dépend de l'interaction de la lumière avec sa surface, mais c'est rarement le cas en synthèse d'images, où des couleurs naturelles sont définies assez arbitrairement. En revanche, l'*aspect* d'un objet peut difficilement être réduit à ce seul paramètre. Le modèle d'éclairement (cf. section 1.2 page ci-contre) requiert des informations supplémentaires, qui peuvent ne pas être représentées géométriquement pour des raisons de coût. Ces informations constituent un ensemble riche appelé *texture*. Une texture peut par exemple servir à représenter les caractéristiques suivantes :

- couleur naturelle de l'objet (caractérisant l'absorption des longueurs d'onde) ;
- micro-géométrie (« relief » de la surface) ;
- indice de réfraction de l'objet sous-jacent ;
- modèle de ré-émission de la lumière (plus ou moins confondu avec les informations précédentes).

## 1.2 Modèles d'éclairement

Un modèle d'éclairement définit la réaction de la lumière rencontrant une surface. En fonction de la complexité des modèles utilisés, les différents effets du comportement de la lumière seront plus ou moins bien reproduits.

Le modèle d'éclairement peut être explicite, ou implicite. « Explicite » signifie que l'on interroge le modèle pour en déduire les transferts d'énergie à appliquer : le modèle guide l'algorithme. « Implicite » signifie que l'on peut avoir choisi un mode de propagation de l'énergie impliquant que le modèle d'éclairement réponde à certains critères : l'algorithme est alors lié au modèle de façon indissociable. C'est le cas de la radiosité (chapitre 2).

Un modèle d'éclairement complexe ne garantit pas forcément qu'un phénomène sera reproduit fidèlement ; cela peut dépendre de la façon dont l'algorithme de rendu utilise ce modèle. Ainsi, il existe un certain nombre de phénomènes connus pour être difficiles à représenter, même s'ils sont prévisibles par le modèle d'éclairement. Ces phénomènes sont donnés en section 1.4.6 page 36.

Enfin, en synthèse d'images, il faut différencier le modèle d'éclairement utilisé et les éventuels post-traitements appliqués lors de la génération d'image et destinés à améliorer le rendu visuel (cf. section 1.3 page 30).

Nous distinguons dans cette section trois modèles classiques d'éclairement : le modèle diffus, le modèle spéculaire, et le modèle utilisant des BRDFs[1].

### 1.2.1 Modèle diffus

Le modèle de ré-émission le plus simple est connu sous le nom de modèle *Lambertien*. Il s'agit du modèle *diffus idéal*. Il prévoit deux choses :

- la lumière est ré-émise de façon isotrope par une surface, dans la demi-sphère supérieure (par rapport à la normale) ;
- la quantité de lumière ré-émise dépend de l'incidence des rayons lumineux ; elle est maximale pour un éclairage zénithal, nulle pour un éclairage rasant.

Le modèle de Lambert s'écrit de la façon suivante :

$$I_d = I_i \times K_{diff} \times \cos\theta$$

- $I_d$ : intensité diffuse, c'est-à-dire l'intensité de lumière sortante (quelle que soit la direction, par isotropie) ;
- $I_i$ : intensité incidente ;
- $K_{diff}$ : paramètre de texture paramétrant le comportement. Un faible coefficient traduit un objet absorbant, tandis qu'un fort coefficient traduit une forte ré-émission ;

[1] *Bidirectional Reflectance Distribution Function*, le terme est expliqué dans la section concernée.

– $\theta$ : angle d'incidence de la lumière, calculé par rapport au vecteur normal à la surface.

Le modèle Lambertien (Figure 1.3) est le modèle d'éclairement minimum pour obtenir une image convaincante dans une scène faisant intervenir des sources lumineuses. Il peut être suffisant si les objets n'ont pas de composante spéculaire (cf. section 1.2.2).

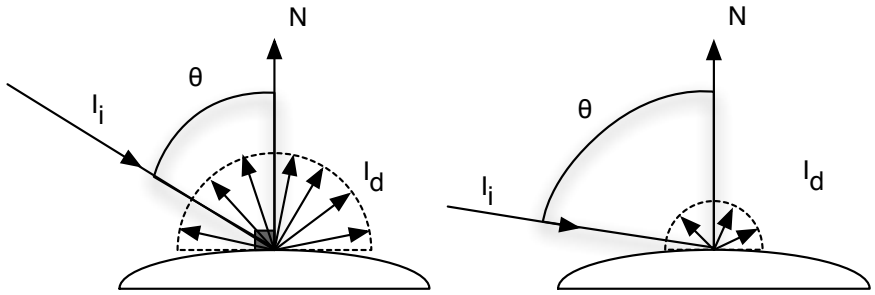



FIG. 1.3 – Le modèle Lambertien : la ré-émission est isotrope, et dépend de l'angle d'incidence ; elle diminue quand l'incidence s'écarte du zénith.

### 1.2.2 Spécularité

La spécularité d'une surface représente sa capacité à se comporter comme un miroir, c'est à dire à ré-émettre la lumière dans une direction privilégiée. En anglais, le terme de *specularity* concerne le miroir idéal, alors qu'on emploie plutôt le terme de *glossy* lorsqu'il y a une certaine diffusion autour du rayon réfléchi. Dans ce manuscrit, nous préférons employer systématiquement le mot *spéculaire*, en précisant « idéal » ou non.

Le miroir est idéalement spéculaire : un rayon incident est réfléchi dans la direction exactement symétrique par rapport à la normale. La propriété de spécularité permet d'observer un polissage sur les objets, caractérisé par des points plus brillants en fonction de l'angle d'observation.

Par rapport au modèle diffus, le modèle spéculaire prend en compte la position de l'observateur pour calculer une image. Les points brillants apparaissent lorsque ce dernier se trouve face à un rayon réfléchi.

Le modèle de Phong (Figure 1.4 page suivante) est un modèle spéculaire classique ; il s'écrit de la façon suivante :

$$I_s = I_i \times K_s \times \cos^n \theta$$

– $I_s$ : intensité spéculaire, c'est-à-dire l'intensité de lumière sortante dans une direction donnée ;
– $I_i$ : intensité incidente ;
– $K_s$ : paramètre de texture paramétrant le comportement. Un faible coefficient traduit un objet à faible composante spéculaire ;

– $\theta$ : angle mesuré entre la direction de sortie considérée et le rayon réfléchi idéal ;
– $n$ : paramètre de texture caractérisant le poli de l'objet. Un $n$ grand traduit une spécularité aiguë, le point brillant se resserrant autour de la direction du rayon réfléchi idéal.

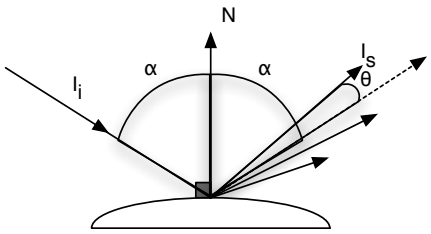



FIG. 1.4 – Le modèle spéculaire de Phong : la ré-émission est plus ou moins intense autour du rayon réfléchi idéal.

### 1.2.3 BRDF : Bidirectional Reflectance Distribution Function

Le modèle diffus et le modèle spéculaire présentés en sections 1.2.1 et 1.2.2 ne sont finalement que des cas particuliers du comportement de la lumière. De façon plus générale, le matériau qui caractérise une surface réagit différemment en fonction des angles d'incidence, et l'émission dépend des angles de sortie. En utilisant les angles polaires $\theta$ et $\phi$ pour caractériser une direction autour d'un point, on peut ainsi définir une fonction de quatre paramètres : $\theta_{in}$, $\phi_{in}$, $\theta_{out}$, $\phi_{out}$, permettant de quantifier le flux d'énergie sortant dans une direction donnée en $(\theta_{out}, \phi_{out})$ par rapport à un flux incident dans une direction donnée en $(\theta_{in}, \phi_{in})$ (Figure 1.5). Cette fonction se nomme BRDF, pour *Bidirectional Reflectance Distribution Function*.

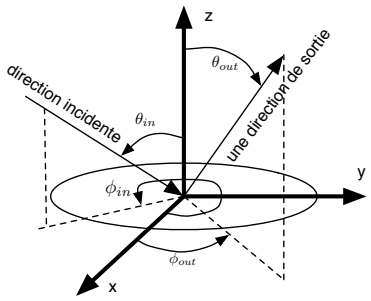



FIG. 1.5 – Les angles polaires comme paramètres d'une BRDF

Comme pour la limite entre géométrie représentable et micro-géométrie modélisée, on peut s'interroger sur l'échelle à laquelle appliquer le principe

de la BRDF. Si l'on représente un arbre, on peut distinguer les BRDFs des feuilles, et de l'écorce. Si en revanche on veut représenter une forêt dans un paysage, il est sans doute judicieux de proposer une BRDF comme réaction globale de la forêt au rayonnement solaire.

Pour associer une BRDF à la représentation d'un objet réel, il peut être utile de la mesurer pour en connaître les caractéristiques, et ainsi la modéliser elle aussi. L'acquisition d'une BRDF pour un objet de taille raisonnable peut se faire avec un dispositif d'éclairage et de mesure permettant de relever les flux d'énergie dans des positions arbitraires ; pour un macro-objet, il faut déployer d'autres techniques d'évaluation [LSW99, SGS+02]. Dans tous les cas, il est probable que l'ensemble des mesures ne soit pas utilisable directement par un algorithme, et qu'il faille passer par une étape de modélisation de la BRDF. Différents outils sont utilisés classiquement pour appréhender ce problème, et ceux-ci sont détaillés dans le chapitre 5 page 117.

L'utilisation d'une BRDF, au prix d'un modèle d'éclairement coûteux, permet d'obtenir un comportement plus réaliste de la lumière dans les images de synthèse. Cependant, cela ne suffit pas encore à représenter tous les comportements de la lumière couramment observables à notre échelle.

### 1.2.4 Limitations classiques

Dans les différents modèles d'éclairement présentés précédemment, la lumière est représentée par des rayons colorés disposant d'une certaine intensité. De plus, la notion de « normale à la surface » est utilisée au point d'impact d'un rayon avec un objet. Or, une analyse plus fine des interactions lumière/objets amène inévitablement à considérer un rayon lumineux comme un transport d'énergie sur différentes longueurs d'onde. En effet, une simple goutte d'eau suffit, par le jeu des réfractions, à séparer un pinceau lumineux en rayons de couleurs différentes. La diffraction remet aussi en cause la notion de point d'impact.

Comme toujours, une analyse plus fine du système réel révèle des comportements plus difficiles à représenter. Souvent, le modèle BRDF peut être étendu pour prendre en compte ces nouveaux éléments. Par exemple, si la BRDF distingue différentes réactions pour différentes longueurs d'onde, elle peut simuler la décomposition de la lumière. Certains phénomènes sont cependant inaccessibles à des modèles d'éclairement trop simples, et sont déportés dans tes traitements algorithmiques spécialisés. De tels phénomènes sont détaillés en section 1.4.6.3 page 37.

### 1.2.5 Vocabulaire et Unités

D'après les différents modèles d'éclairement présentés, il est évident que la façon de représenter la lumière dépend elle-même du degré de précision recherché. Il peut s'agir d'une simple couleur associée à une intensité, ou

d'un spectre d'émission plus large que le visible.

L'une des difficultés à manipuler des modèles complexes est le choix des unités à considérer. Le terme d'« intensité », compris assez intuitivement, peut recouvrir en fait des notions différentes, comme le flux, l'énergie, la radiance, la puissance, la radiosité [FC94].

- l'**Énergie** $E$ ou **Énergie radiante** est exprimée en Joules (J).
- la **Puissance** $P$, ou **Flux radiant**, est l'**Énergie** par unité de temps, mesurée en $J.s^{-1}$. C'est un **flux lumineux**, d'unité équivalent *lumen* (lm).
- l'**Intensité** $I$, ou **Intensité radiante**, est la **Puissance** par unité d'angle solide mesurée en $J.s^{-1}.sr^{-1}$.
- la **Radiance** $L$ est l'**Intensité** par unité de surface projetée de la source, mesurée en $J.s^{-1}.sr^{-1}.m^{-2}$. L'unité de surface projetée est un produit entre la surface infinitésimale émettrice et le cosinus de l'angle d'émission par rapport au zénith. La radiance zénithale est ainsi maximale, et la radiance rasante nulle.
- la **Radiosité** $B$, ou **Émittance** ou **Exitance**, est l'intégration de la **Radiance** sur une sphère englobante. Elle est mesurée en $J.s^{-1}.m^{-2}$.
- l'**Irradiance** $M$ est l'intégration de l'**Intensité** entrante, mesurée comme la **Radiosité** en $J.s^{-1}.m^{-2}$.
- la **Radiance spectrale** $L_\lambda$ est la **Radiance** par longueur d'onde, mesurée en $J.s^{-1}.m^{-3}$

## 1.3 Post-traitement des couleurs

### 1.3.1 Lissage

Un modèle d'éclairement fournit une information à la demande, celle de la lumière émise par un point dans une direction donnée. Si cela est en théorie suffisant pour donner une couleur à chaque pixel de l'image à calculer par un moteur de rendu, il n'en reste pas moins que plus coûteuse est la requête, moins rapide est l'affichage. Selon que l'on cherche à favoriser le temps réel ou la qualité intrinsèque de l'image, cet aspect peut être critique.

En l'occurence, pour optimiser de façon non négligeable l'affichage d'un objet, il est possible de n'évaluer son émission qu'en certains points, et d'interpoler le résultat. Il s'agit donc d'un post-traitement uniquement lié à l'affichage.

#### 1.3.1.1 Le lissage de Gouraud

Dans le cas d'un maillage constitué d'éléments plats, même un modèle d'éclairage complexe comme la BRDF prévoit un aspect quasi uniforme de la surface de chaque élément. Un objet représenté sur cette forme a clairement l'aspect d'un volume taillé grossièrement.

Pour corriger ce problème, on peut s'intéresser non pas à chaque facette, mais à chaque sommet. Le lissage de Gouraud consiste à interpoler l'émission d'un point depuis les sommets de la facette qui le contient (Figure 1.6 page suivante).

#### 1.3.1.2 Le lissage de Phong

Le lissage de Phong ne doit pas être confondu avec le modèle spéculaire de Phong (cf. section 1.2.2 page 26), même s'il s'agit bien du même inventeur. Le lissage de Phong consiste à interpoler non pas les couleurs, mais les normales aux surfaces.

Le coût du calcul de la couleur d'un point dépend en partie du coût d'obtention du vecteur normal en ce point. Cette requête est aisée pour des objets représentés par des équations, mais peut s'avérer complexe pour des objets non primitifs. D'une manière générale, il est alors plus simple de ne calculer les vecteurs normaux qu'aux sommets du maillage, et d'interpoler ces vecteurs à la demande sur le reste de la surface.

Le lissage de Phong et le lissage de Gouraud sont mutuellement exclusifs : ils ne peuvent être employés en même temps. Le premier est plus précis que le second (Figure 1.6 page suivante).

(a) (b) (c)

FIG. 1.6 – (a) En l'absence de lissage, les facettes sont visibles. (b) Un lissage de Gouraud les atténue. (c) Un lissage de Phong les atténue encore mieux.

### 1.3.2 Photométrie : la perception de l'œil humain

Il semble banal de dire qu'une image est destinée à être perçue par l'œil humain. Pourtant, cette simple assertion soulève de nouveaux problèmes de modélisation, dont la particularité est d'être extérieurs aux mondes virtuels pris en charge par un moteur de rendu.

Cela intervient à deux niveaux : tout d'abord, l'image produite doit être compatible avec la perception attendue ; ensuite, elle doit stimuler la rétine de façon adaptée à l'illusion à reproduire, notamment dans les contrastes. La première contrainte est généralement liée à la préservation de l'image si elle passe à travers différents matériels avant d'atteindre la rétine ; il s'agit de corriger les déformations induites par le matériel. La deuxième contrainte consiste à « tricher » sur les couleurs de l'image produite pour simuler un effet par illusion d'optique.

#### 1.3.2.1 Correction des déformations matérielles

La correction des couleurs intervient à plusieurs niveaux dans une chaîne de production, ou de traitement d'image. Chaque appareillage est en effet caractérisé par une fonction de transfert qui modifie un signal d'entrée. Cette fonction de transfert doit être connue pour être corrigée a posteriori. La connaître nécessite d'effectuer des mesures sur l'appareil, avec des sondes et des étalons.

**Correction gamma**

Un exemple simple de correction est la *correction gamma* d'un écran [Poy98]. Les écrans transforment un voltage en intensité lumineuse. Le voltage peut être linéairement déduit du signal d'entrée - un buffer vidéo en composantes rouge, vert, bleu - tandis que l'intensité dépend d'une loi physique reliée à la technologie. Pour un écran cathodique, c'est une loi en puissance :

$$I = (V + \epsilon)^{\gamma}$$

- V est le voltage, I l'intensité
- $\epsilon$ traduit le « point noir »

– $\gamma$ est une constante variant entre 2.35 et 2.55

La « correction gamma » consiste à appliquer une fonction au signal d'entrée pour compenser la déformation induite par le matériel. Il s'agit au final d'adapter les images au mieux à la perception humaine, non linéaire.

**Profils ICC**

De manière générale, chaque appareil intervenant en traitement d'image est caractérisé par l'ensemble des couleurs qu'il peut reproduire (le *gamut*), et sa fonction de transfert. Une fois ces données mesurées, elle peuvent être stockées en un format standardisé de l'International Color Consortium, nommé *profil ICC*. Le profil ICC permet de tenir compte des déformations de façon logicielle dans la chaîne de traitement.

#### 1.3.2.2 Codage et Tone mapping

La correction des déformations matérielles vise à *préserver* l'information. Il est également possible de modifier l'information dans un but précis.

**Codage**

La rétine humaine est plus sensible aux nuances dans le sombre que dans le clair. Certaines teintes sont également mieux discriminées à l'œil nu. La connaissance des ces imperfections permet d'optimiser le codage d'une image, en privilégiant les zones sensibles, c'est-à-dire en leur octroyant plus de bits pour accroître leur précision [JJS93].

**Tone mapping**

L'intervalle des intensités perçues pour une image dépend des capacités du matériel d'affichage. La photographie d'un soleil n'éblouit pas qui la regarde. Appliquer des changements d'échelle linéaires à une image naturellement contrastée provoque des zones de sous-exposition ou de surexposition en fonction de l'échelle choisie. La mise au point d'opérateurs capables de donner à une image l'illusion du contraste naturel est à lui seul un domaine de recherche. Il s'agit des opérateurs de *tone-mapping* [TR93, War94, PTYG00, LRP97]. Les opérateurs de tone-mapping peuvent apporter des modifications locales aux pixels en fonction de leur environnement, ou encore de tenir compte du temps d'adaptation de la rétine dans les changements brusques d'intensité.

## 1.4 Techniques de rendu élémentaires

Les modèles géométriques et d'éclairement étant définis, un algorithme doit être capable de produire une image en interagissant avec eux. Ce sont les techniques de rendu. En fonction de l'effet recherché - rapidité de calcul ou réalisme de l'image - on privilégiera l'une ou l'autre technique.

Le chapitre 2 étant consacré à la description de techniques de rendu avancées, nous ne présentons ici que des algorithmes élémentaires permettant d'introduire les problématiques usuelles. Il s'agit de la technique de projection, et du lancer de rayons (ray-tracing) « élémentaire ». Il existe des versions plus complexes du lancer de rayons, présentées elles aussi en section 2.2.

### 1.4.1 La caméra

La géométrie d'une scène, les propriétés de ses surfaces, le modèle d'éclairement, étant définis, un moteur de rendu doit être capable de calculer une image de la scène. Calculer une image signifie en pratique simuler ce que verrait une caméra ou un observateur positionné dans la scène. Notons que cet observateur est absent de la scène ; il n'a aucune interaction avec ses constituants, il s'agit simplement d'une vue de l'esprit pour projeter une image. L'observateur n'est pas non plus limité à l'œil humain ; la caméra dispose elle-même d'un modèle de projection et de déformation, qui peuvent s'éloigner d'un modèle classique.

### 1.4.2 Inadéquation du modèle photographique

Pour calculer l'image qu'afficherait une caméra, la technique la plus intuitive, mais qui s'avère impraticable dans un premier temps, est celle du modèle photographique. Dans un tel modèle, c'est l'aspect corpusculaire de la lumière qui serait pris en compte. Les sources lumineuses émettent des photons, qui se propagent en ligne droite en l'absence d'interactions. Lorsqu'un photon rencontre un objet, le point d'impact et son modèle d'éclairement définissent le comportement à adopter : absorption ou ré-émission. Lorsqu'un photon arrive sur la pellicule de la caméra, il y imprime une information. Calculer l'image reviendrait alors à compter les photons apparaissant sur la pellicule virtuelle et à en déduire les couleurs.

En synthèse d'images, une implémentation naïve de cette technique serait prohibitive à l'échelle considérée : il est hors de portée de simuler chaque photon nécessaire à impressionner l'ensemble d'une pellicule, même pour un temps de pose infime. En revanche, d'excellents résultats peuvent être obtenus, d'une part avec des techniques d'échantillonnage, d'autre part avec un modèle de déplacement instantané des photons sous forme de rayons rectilignes. Ce modèle est à la base de la technique du *lancer de rayons*. Les

interactions des photons peuvent être simulées en suivant des lois probabilistes. Ce sont des méthodes dites de *Monte-Carlo*. De telles méthodes sont détaillées en section 2 page 39.

### 1.4.3 Projection et algorithme du peintre

La plupart des visualiseurs orientés temps réel utilisent des techniques de projection pour afficher la géométrie, et l'interpolation bilinéaire pour le rendu des couleurs. La caméra étant représentée par un écran, il est possible de calculer la projection 2D de chaque objet de la scène sur (ou hors de) cet écran. Lorsque le modèle géométrique favorise les maillages, la projection des sommets suffit à obtenir des informations suffisantes.

Pour reproduire l'effet de profondeur, de telle sorte que les parties cachées des objets restent masquées sur l'image, on utilise un algorithme appelé « algorithme du peintre[1] ». Ce dernier consiste à peindre sur l'écran les objets les plus éloignés d'abord, pour qu'ils soient recouverts ensuite par les objets les plus proches. Cet algorithme peut être mis en défaut, car deux objets peuvent s'entrecroiser sans possibilité de dire que l'un est derrière l'autre. Heureusement, les cartes graphiques offrent une autre solution plus efficace permettant de stocker au moins deux informations par pixel : sa couleur et la distance au point d'observation. En peignant les projections des objets dans un ordre quelconque, on peut ainsi décider pixel par pixel s'il faut recouvrir l'ancienne couleur ou non.

Basiquement, la couleur des pixels est calculée par un modèle d'éclairage simple, en illumination « directe » uniquement, c'est-à-dire qui ne calcule aucune ré-émission. Un modèle de Phong et un lissage de Gouraud sont des standards supportés par la plupart des cartes graphiques.

### 1.4.4 Lancer de rayons élémentaire

Les lois de la réflexion et de la réfraction étant réversibles, c'est-à-dire respectant le principe du trajet inverse de la lumière, l'idée principale du lancer de rayons « élémentaire » est de remonter le long des rayons atteignant la caméra. En première approximation, cela permet de limiter les calculs aux seules informations utiles à la constitution de l'image finale.

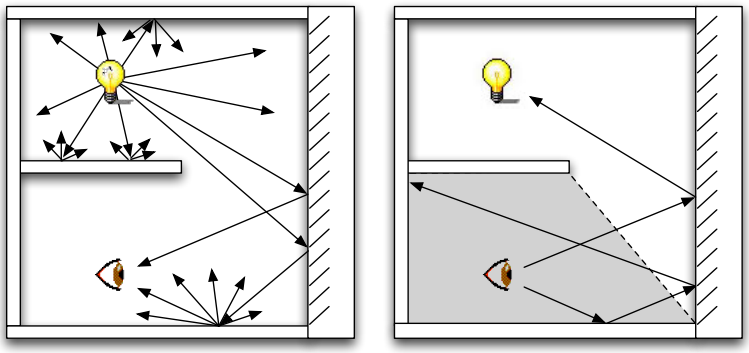
Remonter le long d'un rayon $R$ signifie partir de son extrémité (un pixel de l'image) et trouver son origine dans la scène ; puis de cette origine remonter le long du rayon réfléchi $R'$ ou réfracté $R''$ qui a pu générer $R$. Si l'on remonte ainsi jusqu'à une source lumineuse, alors on en déduit rétrospectivement l'énergie transportée. En limitant ainsi le nombre de rayons à explorer, on évite les calculs inutiles sur des objets non visibles.

La probabilité de trouver une source lumineuse sur un chemin optique est faible. L'éclairage est donc scindé en deux types pour compenser cela :

[1] à tort, un peintre n'applique pas forcément cette technique

l'éclairage direct et l'éclairage indirect. À chaque point d'impact, on peut tester s'il existe un chemin direct vers une source lumineuse pour en capter l'énergie sans avoir à la découvrir « par hasard ». Les rayons relancés selon les lois de l'optique géométrique pistent quant à eux un éclairage indirect.

Cette approche échoue toutefois à représenter tous les rayons utiles, car l'hypothèse que seules les sources lumineuses contribuent à l'éclairage direct s'oppose à la prise en compte des réflexions diffuses. L'ensemble des rayons utiles pour l'image est donc largement sous-évalué et certaines zones de l'image ne seront pas éclairées comme dans un système réel (Figure 1.7)



(a) (b)

FIG. 1.7 – (a) Le miroir (mur de droite) provoque un éclairage indirect permettant à l'observateur de distinguer les murs qui l'entourent. (b) Les seuls rayons lancés échouent à détecter cet éclairage et la zone de l'observateur reste dans l'ombre.

#### 1.4.4.1 Rayons et intersections

En l'absence d'interactions, la lumière se propage en rayons rectilignes. En l'absence de milieu participatif ou non homogène, les interactions sont donc limitées aux surfaces des objets. Dans ce cas, seuls les points d'impact des rayons, c'est-à-dire leurs intersections avec les surfaces du modèle géométrique, sont à considérer.

La complexité d'un tel calcul d'intersections dépend de la complexité du modèle géométrique choisi, ainsi que de celui du rayon, même si le plus souvent celui-ci est assimilé à une simple droite mathématique idéale. L'ensemble des intersections d'une droite avec une surface implicite (cf. section 1.1.3 page 22) est sensiblement plus difficile à calculer que celui d'une droite avec un maillage de triangles (cf. section 1.1.5 page 23).

L'ensemble des primitives d'un moteur de rendu basé sur le lancer de rayons est donc en théorie limité aux primitives pour lesquelles on dispose d'un algorithme efficace de calcul d'intersection. Il existe de tels algorithmes pour les surfaces implicites [MKW+04, Raf].

Le calcul des intersections est connu pour être la pierre d'achoppement du lancer de rayons représentant environ 90% du total des calculs. Avec une batterie d'optimisations et une architecture dédiée, le lancer de rayons en temps-réel est cependant devenu une réalité, même sur des scènes extrêmement complexes [WPS+03].

### 1.4.5 Illumination globale

L'illumination globale n'est pas un algorithme, mais une approche particulière du problème de l'éclairement dans une scène, qui peut être appréhendée par de nombreux algorithmes différents. On peut en distinguer deux grandes classes : ceux qui utilisent le lancer de rayons (section 2.2 page 46), et ceux qui utilisent des techniques d'éléments finis (section 2.3 page 50). La résolution de la radiosité dérive de ce problème. Nous consacrons le chapitre 2 page 39 à la présentation de ces algorithmes.

### 1.4.6 Difficultés classiques

Chaque technique de rendu a ses faiblesses dans la reproduction des comportements observables d'un système réel. Il a par exemple été mentionné que le lancer de rayons s'accommode mal des réflexions diffuses. Il est souvent possible de pallier ces faiblesses algorithmiquement, en greffant des extensions aux modèles de calcul. Un miroir peut par exemple être simulé en dédoublant la géométrie d'une pièce. Ces extensions, si efficaces qu'elles soient, sont souvent des artifices spécialisés dans la représentation d'un comportement particulier du système réel. Cette section présente certains de ces comportements particuliers qui peuvent faire l'objet d'extensions particulières des moteurs de rendu.

#### 1.4.6.1 Sources lumineuses

La notion de source lumineuse, qui définit l'origine de tout éclairage en image de synthèse, est une première difficulté. En plus de la couleur et de l'intensité, on peut exiger qu'elle ait une direction d'émission naturelle. Sa taille pose également un problème de modélisation : le lancer de rayons fonctionne généralement avec des sources ponctuelles (non étendues), qui génèrent des ombres tranchées. La prise en compte de sources étendues peut se faire avec un échantillonnage (Figure 1.8 page suivante) pour générer des ombres plus douces.

#### 1.4.6.2 Caustiques

La convergence des rayons lumineux réfléchis sur une surface courbe produit une enveloppe lumineuse caractéristique, appelée *caustique* (Figure 1.9 page ci-contre). Il s'agit souvent d'un détail à petite échelle par rapport à

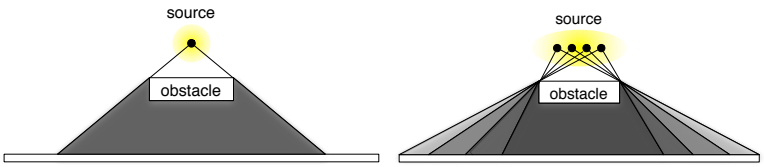



FIG. 1.8 – (a) Source ponctuelle. (b) Source étendue implémentée par un ensemble de sources ponctuelles : des zones de pénombre apparaissent.

la scène, que l'on peut déporter dans un algorithme spécifique, tel le *photon mapping* (cf. section 2.2.6 page 47) [GWS04].

FIG. 1.9 – Une caustique dans une tasse à café

#### 1.4.6.3 Irisation, Fluorescence

Parmi les phénomènes observables dus à la répartition de la lumière sur un spectre, on peut citer l'irisation [Sta99] (décomposition de la lumière, Figure 1.10), la fluorescence [Gla94, WTP01] (décalage de longueur d'onde pour la ré-émission).

FIG. 1.10 – Irisation à la surface d'un CD

#### 1.4.6.4 Polarisation, Interférences, Diffraction

Les modèles d'éclairement font rarement une interprétation ondulatoire de la lumière. Le principe d'Huygens prévoit pourtant les phénomènes de diffraction et d'interférences, observables à travers les fentes d'Young. La polarisation repose elle aussi sur l'aspect ondulatoire de la lumière. La représentation de ces phénomènes peut faire l'objet d'algorithmes spécifiques [WTP01].

#### 1.4.6.5 Dispersion sous-surfacique

La dispersion sous-surfacique (*sub-surface scattering*) représente le fait que les photons puissent pénétrer légèrement sous la surface d'un objet et être ré-émis à proximité du point d'impact. On peut prendre cela en compte dans un modèle de BRDF [HK93, JMLH01].

#### 1.4.6.6 Milieux participatifs

Dans un milieu dit « participatif », les photons ont une probabilité d'interaction avec le milieu, et le modèle de rayon n'est plus adapté. Les collisions peuvent provoquer aussi bien une dispersion (éclatement) qu'une convergence. Le modèle d'illumination doit être reconsidéré pour prendre en compte ces modifications [JC98, PPS97].

#### 1.4.6.7 Temporalité

Les algorithmes de rendu considèrent souvent la répartition de la lumière dans une scène comme un état à un instant donné. Or, cet état est calculé d'après les multiples ré-émissions de la lumière. Il s'agit en fait d'une approximation puisque les photons ont une vitesse finie. Cependant, prendre en compte la vitesse de la lumière n'est pas forcément nécessaire pour reproduire certains effets. Par exemple, l'indice de réfraction d'un milieu utilisé lors de la traversée d'un dioptre est une conséquence du changement de vitesse des photons, mais les lois de Snell-Descartes représentent cela sans calculer ces vitesses.

Négliger la vitesse de la lumière provoque toutefois un résultat légèrement faux sur l'évaluation de la quantité d'énergie présente dans la scène. Un chemin optique fermé accroît artificiellement cette quantité, puisque l'on additionne alors des photons à eux-mêmes. En pratique, cette erreur est souvent négligeable.

L'absence de temporalité interfère également avec la notion de *phosphorescence*, un objet pouvant accumuler l'énergie lumineuse pour la ré-émettre plus tard. L'ajout d'une dimension temporelle dans un modèle d'émission permet de simuler cela [Gla94].

# Chapitre 2

# Méthodes de calcul d'illumination globale

La section 1.4.5 page 36 a positionné brièvement, dans le domaine de la synthèse d'images, le problème de l'illumination globale. Dans le présent chapitre, nous le détaillons. Celui-ci repose sur des équations physiques décrivant la conservation de l'énergie. Les inconnues de ces équations sont les données d'émission nécessaires à rendre une image. La théorie sous-jacente est donc à la fois physique puisqu'il s'agit de décrire un système réel, et mathématique dans la manipulation de ces équations pour en évaluer une solution. Un bon tour d'horizon des méthodes de résolution utilisables peut se trouver en [DBB03] et [FC94]. Le présent chapitre se veut un résumé de ces techniques. Il constitue en lui-même une introduction à nos travaux sur une nouvelle technique (chapitre 4) en évoquant les astuces habituellement déployées, et les capacités intrinsèques des modèles, dont nous avons pu nous inspirer.

## 2.1 Illumination globale et Radiosité

### 2.1.1 L'équation d'illumination globale

Le chapitre 1 présente les problématiques courantes rencontrées en synthèse d'images. Les algorithmes les plus simples ont été présentés pour illustrer les étapes qui jalonnent un processus de génération d'image. Pour obtenir le résultat le plus réaliste possible, il est toutefois nécessaire de disposer d'une formalisation avancée de la distribution de la lumière dans une scène. Une telle formalisation existe, sous forme d'une équation appelée *équation d'illumination globale*.

Cette équation traduit l'équilibre énergétique dans une scène (sans considérer de dimension temporelle, cf. section 1.4.6.7 page 38), et s'exprime avec la *Radiance* (cf. section 1.2.5 page 28). Son interprétation est que l'émission d'un élément est la somme de son émission propre, et de la ré-émission de l'énergie qu'il reçoit. L'inconnue de cette équation est la radiance $L$ en tout point $x$, pour toute direction sortante $(\theta_{out}, \phi_{out})$

$$L(x, \theta_{out}, \phi_{out}) = \underbrace{L_e(x, \theta_{out}, \phi_{out})}_{\text{émission propre}} + \underbrace{\int_\Omega \rho_{bd}(x, \theta_{in}, \phi_{in}, \theta_{out}, \phi_{out}) L_i(x, \theta_{in}, \phi_{in}) \cos\theta_{in} d\omega}_{\text{ré-émission}} \tag{2.1}$$

Les différents termes ont les significations suivantes :

- $x$ : point d'émission considéré ;
- $L$ : radiance du point $x$ (en $J.s^{-1}.sr^{-1}.m^{-2}$) ;
- $\theta_{in}$, $\phi_{in}$ : angles polaires, dans un repère local à $x$, de la direction d'incidence ;
- $\theta_{out}$, $\phi_{out}$ : angles polaires, dans un repère local à $x$, de la direction de sortie dont on veut calculer la radiance ;
- $L_e$ : radiance d'émission propre à $x$ s'il appartient à une source lumineuse ; cette fonction est connue ;
- $L_i$ : radiance incidente en $x$, dépendant des radiances des autres points de la scène ;
- $\Omega$ : sphère d'intégration centrée en $x$ ;
- $\rho_{bd}$ : la BRDF ; cette fonction est un ratio entre une radiance sortante et un flux de radiance entrante. La BRDF est supposée connue en tout point $x$, pour toutes directions.
- $\cos\theta_{in}$ : terme traduisant l'*unité de surface projetée* (cf. 1.2.5 page 28) ;
- $d\omega$ : intégrande angle solide.

La cohérence de l'équation est assurée par la BRDF $\rho_{bd}$. Elle doit assurer qu'aucune énergie n'est créée à chaque ré-émission.

### 2.1.2 Calcul d'une image

L'équation d'illumination globale et l'équation de radiosité qui en découle (section 2.1.3) sont indépendantes de toute visualisation : elles ne font que traduire la répartition de l'énergie dans un état stable. Elles sont donc remarquables en cela qu'une fois résolues, la visualisation ne nécessite plus aucun calcul d'éclairement : obtenir une image revient à interroger les résultats stockés pour déduire la couleur de chaque pixel. La temporalité étant négligée (cf. section 1.4.6.7 page 38), une image n'est qu'une vision de la scène prise dans un état donné. Déplacer la caméra ne remet pas en cause les calculs déjà effectués (Figure 2.1).

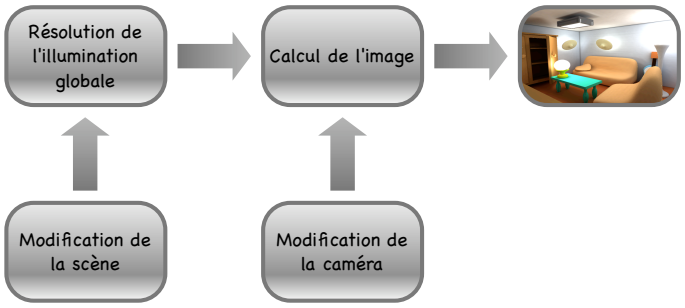



FIG. 2.1 – Si l'équation d'illumination globale est résolue, le rendu de l'image peut être dissocié des calculs d'éclairage.

Cependant, résoudre l'équation signifie aussi stocker sa solution. Les algorithmes ne contiennent pas forcément l'intégralité de la solution en mémoire, et la stocker peut s'avérer coûteux. De ce fait, les algorithmes de résolution de l'équation d'illumination globale ne décorrèlent pas forcément la visualisation des calculs d'éclairement comme dans la figure 2.1. Les techniques de lancer de rayons, notamment, (cf. section 2.2 page 46) ne résolvent pas forcément l'équation au sens analytique du terme, et les calculs ne sont effectués que pour un emplacement donné de la caméra.

Dans le cas d'une animation image par image, se pose la question de ne pas recalculer entièrement une solution, mais d'être capable de la mettre à jour de façon incrémentale [ABM01, BMA03].

### 2.1.3 De l'illumination globale à la radiosité

Résoudre le problème de l'illumination globale est souvent simplifié en la résolution de la seule *radiosité*, laquelle représente l'intégration de la radiance sur une sphère englobante (cf. section 1.2.5 page 28). En conséquence, un calcul de radiosité réduit généralement le modèle d'éclairement au modèle diffus idéal, et néglige les comportements complexes que représentent les BRDFs (cf. section 1.2.3 page 27).

La radiosité est une solution intéressante pour un rendu global convaincant, mais se révèle souvent inefficace pour représenter des détails à petite échelle. De plus, le modèle diffus idéal prive la radiosité des effets spéculaires, aussi courants que la réflexion et la réfraction. Combiner un calcul de radiosité avec un calcul de ray-tracing permet de bénéficier des atouts de chacun : l'éclairement indirect pour le premier, et les comportements optiques précis des rayons pour le deuxième. Cette approche mixte peut donc donner des résultats visuellement très satisfaisants.

#### 2.1.3.1 Simplification de l'équation d'illumination globale

Dans un premier temps, on peut considérer l'équation d'illumination globale comme une description trop complexe du système. En se plaçant dans le cadre d'un modèle d'éclairement uniquement diffus sur des objets opaques, cette équation peut être simplifiée.

Cette simplification a lieu sur les quantités émises, qui peuvent être intégrées sur l'ensemble des directions sans perte d'information, sur les fonctions BRDF qui sont alors uniformes, et sur la notion de radiance incidente qui est liée à la géométrie de la scène. Ces trois étapes sont détaillées ci-après.

#### 2.1.3.2 Intégration de la radiance émise

La diffusion idéale est caractérisée par des ré-émissions isotropes. La direction d'émission n'est plus discriminante, en ce sens que $L(x, \theta, \phi) \equiv L(x)$ et l'on peut travailler sur des quantités intégrées sur une sphère englobante $\Omega$. La radiosité est effectivement l'intégration de la radiance dans toutes les directions (équation 2.2).

$$\begin{aligned} B(x) &= \int_{\Omega} L(x, \theta, \phi) \cos\theta d\omega \\ &= \int_{\Omega} L(x) \cos\theta d\omega \\ &= L(x) \int_{\Omega} \cos\theta d\omega \\ &= \pi L(x) \end{aligned} \tag{2.2}$$

#### 2.1.3.3 Simplification des BRDFs

De même, on peut définir le coefficient de réflexion diffuse $\rho_d(x)$ traduisant l'uniformité de la BRDF (équation 2.3).

$$\rho_d(x) = \pi \rho_{bd}(x, \theta_{in}, \phi_{in}, \theta_{out}, \phi_{out}) \tag{2.3}$$

#### 2.1.3.4 Interprétation de la radiance incidente

Dans un milieu non participatif où la lumière se propage en ligne droite entre les dioptres, et en supposant tous les objets opaques, La radiance incidente dans la direction absolue $\overrightarrow{d}$, notée $L_i(x, \overrightarrow{d})$, est en réalité la radiance sortante de l'unique point $y$ visible de puis $x$ dans la direction -$\overrightarrow{d}$. Si $\overrightarrow{d}$ est décrite dans le repère local à $x$ par les angles polaires $\theta$ et $\phi$, et que -$\overrightarrow{d}$ est décrite dans le repère local à $y$ par les angles polaires $\theta'$ et $\phi'$ (Figure 2.2), on peut écrire

$$L_i(x, \theta, \phi) = L(y, \theta', \phi') = \frac{B(y)}{\pi} \tag{2.4}$$

Le facteur d'angle solide $d\omega$ peut alors être interprété par l'angle solide généré par un élément $y$, d'aire $dy$, sur $x$, soit

$$d\omega = \frac{\cos\theta' dy}{||x - y||^2} \tag{2.5}$$

Enfin, pour permettre l'écriture d'une intégrale sur tous les points de la scène, on peut caractériser le point $y$ visible depuis $x$ par une fonction binaire de visibilité $V(x, y)$ telle que

$$V(x, y) = \begin{cases} 1 & \text{si } y \text{ est visible depuis } x \\ 0 & \text{sinon} \end{cases}$$

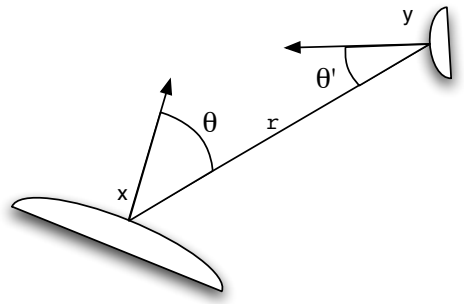



FIG. 2.2 – La radiance incidente en $x$ depuis $y$ dépend de l'angle solide généré par $y$ sur $x$, donc de la distance et de l'orientation de $y$.

#### 2.1.3.5 Formulation de l'équation de radiosité

En combinant les manipulations précédentes, l'équation d'illumination globale (2.1) peut être transformée pour devenir l'équation (2.6) d'inconnue $B(x)$ :

$$B(x) = B_e(x) + \rho_d(x) \int_{y \in \text{ scène}} B(y) \frac{\cos\theta \cos\theta'}{\pi \ r^2} V(x,y) dy \qquad (2.6)$$

- $B_e(x)$ est la radiosité propre de l'élément en tant que source lumineuse, elle est connue ;
- $\rho_d$ paramètre la capacité de ré-émission d'un élément ;
- $r = ||x - y||$, distance entre les éléments ;
- $\cos\theta$ traduit l'importance de l'incidence dans le modèle d'éclairement diffus ;
- $\frac{\cos\theta'}{r^2}$ traduit l'influence de $y$ sur $x$, qui est d'autant plus faible que l'élément $y$ est lointain et « penché ».

### 2.1.4 Résolution analytique de l'équation

L'équation d'illumination globale est énoncée mathématiquement. Si elle est cohérente, une unique solution existe. Il s'agit en effet d'un problème de point fixe dans l'espace des fonctions intégrables. La solution devrait pouvoir être énoncée analytiquement. Cependant, même des cas très simples, comme deux bandes perpendiculaires de largeur connue et de longueur infinie, peuvent conduire à des solutions non explicites [FC94].

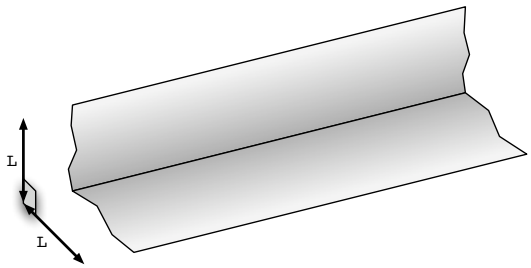



FIG. 2.3 – Deux bandes perpendiculaires de largeur connue et de longueur infinie, ne permettent pas de calculer explicitement une solution de radiosité.

L'équation de radiosité (2.6) s'interprète mathématiquement comme une *équation intégrale de Fredholm du second type*, de la forme :

$$\phi(x) = f(x) + \lambda \int_{-\infty}^{\infty} K(x,t)\phi(t)dt \qquad (2.7)$$

On sait exprimer la solution d'une telle équation comme une *suite de Neumann*. Pour information, une suite de Neumann exprime l'inverse de $(1-X)$, dans un espace de Banach, sous forme de série convergente. On écrit ainsi :

$$(1-X)^{-1} = \sum_{k=0}^{\infty} X^k$$

Le noyau d'intégration $K$ est cependant peu adapté aux calculs formels dans le cas de la synthèse d'images, puisque la fonction de visibilité $V(\cdot, \cdot)$ le prive de continuité.

Il existe plusieurs algorithmes approchés pour résoudre le problème de l'illumination globale. La plupart sont des algorithmes basés sur l'échantillonnage et le lancer de rayons, et sont présentés en section 2.2 page suivante. Dans le cas de la radiosité, les techniques sont plus analytiques et font appel aux notions d'éléments finis, présentés en section 2.3 page 50.

## 2.2 Algorithmes par lancer de rayons

Le principe du lancer de rayons peut être appliqué aux calculs d'illumination globale. L'interprétation de l'équation est immédiate : chaque rayon transporte une certaine radiance, et lors d'un impact, si l'on est dans le sens des photons (light-tracing), on peut échantillonner les rayons sortants ; si l'on est dans le sens inverse des photons (ray-tracing), on échantillonne les rayons qui auraient pu générer celui dont on vient ; cela permet de capter les éclairages indirects.

Si le ray-tracing simple échoue à représenter une grande partie des rayons utiles, c'est justement l'échantillonnage qui permet de corriger le problème, au prix, bien sûr, d'une complexité accrue. La difficulté réside dans un choix de l'échantillonnage, qui doit être efficace et pertinent tout en restant raisonnable.

Dans ce contexte, deux approches de lancer de rayons peuvent être utilisées : soit depuis les sources lumineuses (light-tracing), soit depuis la caméra (ray-tracing). Les deux approches peuvent aussi être combinées (bidirectional ray-tracing).

### 2.2.1 Ray-tracing stochastique

Dans le ray-tracing simple (section 1.4.4 page 34), chaque impact de rayon génère au plus deux nouveaux rayons à explorer : un rayon réfléchi et un rayon réfracté, selon les lois de Snell-Descartes. Si l'on échantillonne, on peut créer autant de nouveaux rayons à explorer que désiré, dans de multiples directions. Cela crée un arbre de rayons, dont la racine est la caméra. Si l'exploration est fructueuse sur les branches qui véhiculent une énergie importante, il convient en revanche d'élaguer les autres.

La mise en œuvre de cette technique repose sur l'échantillonnage choisi pour relancer les rayons. Différentes astuces peuvent être déployées pour diminuer le bruit, en échantillonnant de façon non uniforme, et éventuellement dynamique, pour privilégier les zones sensibles au fur et à mesure qu'on les découvre.

### 2.2.2 Light-tracing

En partant de la caméra, l'idée du ray-tracing élémentaire est de ne calculer que des rayons utiles. Dans le contexte de l'échantillonnage, cette assertion est moins valable, et il n'est pas forcément moins efficace de lancer les rayons depuis les sources en espérant qu'ils atteignent la caméra ; c'est le principe du light-tracing.

En ray-tracing, à chaque impact, on teste la présence ou non d'un éclairage direct en vérifiant si les sources lumineuses sont visibles depuis ce point. Cela force la découverte des sources lumineuses pour le rayon cou-

rant (cf. section 1.4.4 page 34). En light-tracing, la même idée est utilisée pour forcer la découverte de la caméra. À chaque point d'impact, on vérifie s'il existe un rayon direct vers cette dernière. Cela permet de maximiser le nombre de chemins optiques atteignant la caméra. Malheureusement, celle-ci n'est pas ponctuelle, en ce sens que l'image à calculer est constituée de nombreux pixels. Un rayon atteignant la caméra n'a donc d'influence que sur quelques-uns de ces pixels. En light-tracing, on n'a pas la garantie que l'ensemble des pixels de l'image reçoive des rayons.

### 2.2.3 Bi-directional path-tracing

Le bi-directional path-tracing consiste à combiner ray-tracing et light-tracing pour profiter des avantages des deux. Des rayons sont tracés depuis le caméra et depuis la source. En connectant les chemins générés, on favorise les chemins importants pour l'image. [LW94, Vea94]

### 2.2.4 Modèle Metropolis

Le modèle de transport de lumière « Metropolis » se base lui aussi sur un lancer de rayons, mais l'exploration de l'espace est assez particulière. Les rayons ont une probabilité de mutation, de séparation et de recollement, dépendant de l'importance des rayons obtenus dans l'image finale. Cela permet d'explorer de nombreux chemins optiques et de capturer des phénomènes locaux comme les caustiques. [VG97]

### 2.2.5 Irradiance cache

L'irradiance cache [WRC88] n'est pas une technique à proprement parler mais une optimisation du ray-tracing. Avec le lancer de rayons, il est difficile de prévoir où auront lieu les points d'impact. Or, il peut se faire que deux chemins optiques se rejoignent. Dans ce cas, il serait souhaitable de ne pas dupliquer les calculs de l'arbre d'exploration.

Pour y remédier, il est envisageable de stocker dans une structure adaptée les résultats obtenus aux points d'impact déjà explorés. Un *octree* peut convenir. Précisons toutefois que la probabilité pour deux chemins optiques de tomber *exactement* au même point d'impact est faible. Il est donc prévu dans l'algorithme d'*irradiance cache* de pondérer les données stockées des points d'impact avoisinants [WH92].

### 2.2.6 Photon mapping

Le photon mapping [Jen96] est assez proche de l'irradiance cache. Cette technique s'applique cependant au light-tracing plutôt qu'au ray-tracing, et nécessite deux phases distinctes : une phase de propagation et une phase de rendu. Cela s'inscrit dans le schéma de la Figure 2.1 page 41.

La première passe, de light-tracing (rayons tracés en partance des sources), permet de stocker aux différents points d'impact la densité de photons reçus. Ces informations constituent la *photon map*, souvent implémentée par une structure de *kd-tree*. La deuxième passe est un ray-tracing élémentaire qui interroge la photon-map au lieu des seules sources lumineuses pour calculer la couleur des pixels, de façon modulée comme avec l'irradiance cache. Cela permet de considérer un niveau d'éclairage indirect.

Pour capturer des phénomènes très locaux comme les caustiques, il est possible de calculer simultanément plusieurs photon-maps, restreintes à certaines zones. Les caustiques sont très souvent dues à l'éclairage direct d'un objet spéculaire. Il est donc assez facile de repérer des zones susceptibles d'en contenir. Il existe des implémentations temps réel du photon-mapping [GWS04].

### 2.2.7 Instant Radiosity

L'algorithme *Instant radiosity* [Kel97] est une extension du photon-mapping. Au lieu de limiter le light-tracing à une seule passe, l'algorithme prévoit d'en exécuter plusieurs successivement, mais en considérant que chaque point de la photon-map ayant reçu quelques photons se comporte à son tour comme une source lumineuse. Plusieurs passes conduisent donc à un éclairage indirect réaliste. Une image peut être calculée après chaque passe, ce qui en fait un algorithme incrémental permettant des raffinements successifs.

### 2.2.8 Random Walk

La technique du Random Walk diffère des méthodes de lancer de rayons précédentes car elle s'applique sur un maillage de la scène. Plutôt que de découvrir des points d'impact dans la scène, on la pré-découpe en éléments dont on va calculer la radiosité moyenne. De cette façon, l'ensemble des inconnues à évaluer est pré-déterminé puisqu'il s'agit des $n$ radiosités des $n$ éléments. Sur cette base, le Random Walk reste une méthode probabiliste avec échantillonnage de rayons, contrairement aux méthodes d'éléments finis présentées en section 2.3 page 50.

Le principe utilisé dans le Random Walk est assez proche du photon-mapping (section 2.2.6 page précédente) qui accumule des photons en certains endroits. On part du principe que chaque élément de surface donne naissance à des photons avec une certaine probabilité dépendant de son émittance propre. Le photon est ensuite envoyé dans une direction aléatoire. La probabilité de tomber dans un élément très visible depuis la source est plus grande que celle de tomber dans un élément peu visible ; la variable aléatoire responsable de la direction est donc choisie pour que cette probabilité de transition soit égale aux facteurs de forme. Enfin, après une telle transition, le photon a une probabilité d'absorption ou de ré-émission en fonction de la réflectivité de l'élément qui l'a reçu. Chaque élément doit

compter combien de photons il reçoit (collision), absorbe (absorption) ou retransmet (survie). Un estimateur bien choisi permet alors d'interpréter ce nombre en termes de radiosité [DBB03].

## 2.3 Algorithmes par éléments finis

Les algorithmes par lancer de rayons seuls doivent procéder par échantillonnage. Cela revient à discrétiser, pour chaque point, l'ensemble des directions à considérer. Une autre approche essentielle de la résolution de l'illumination globale repose sur un maillage des surfaces en *patches*, il s'agit alors d'une discrétisation de la scène.

En discrétisant les surfaces, on fait l'hypothèse que la radiance (ou la radiosité selon le cas) peut être représentée par une fonction simple sur chaque patch. Généralement, on impose à la radiosité d'y être constante. Si la radiosité est calculée pour chaque élément, elle se propage alors à tous les points de ces éléments. Les transferts d'énergie appliqués au cours de la résolution peuvent alors se faire plus systématiquement d'élément à élément, sans échantillonner de directions particulières.

L'hypothèse de représenter la radiosité comme une fonction sur chaque élément est forte, car rien ne garantit que la solution du système réel soit effectivement représentable avec les fonctions choisies. Une solution de l'équation discrétisée n'est donc pas forcément une solution du système réel. Malgré tout, elle reste satisfaisante, car l'équation d'illumination est une équation très stable. C'est l'approche habituelle des calculs d'éléments finis.

### 2.3.1 Discrétisation

La discrétisation des surfaces consiste à les représenter sous forme de maillage (cf. 1.1.5 page 23), et à proposer un modèle paramétré de radiosité sur chaque élément. Très souvent, on considère la radiosité comme étant constante sur un élément. Si l'on arrive à calculer l'influence des éléments les uns sur les autres, cela permet de représenter le problème comme un ensemble d'équations, qui une fois résolu caractérise l'ensemble du système.

Le maillage considéré pour le modèle d'illumination n'est pas forcément le même que celui du modèle géométrique sous-jacent. En effet, si un mur rectangulaire peut être représenté par deux triangles, un éclairage non uniforme de ce mur nécessite une discrétisation plus fine pour représenter les variations de luminosité. La constitution d'un maillage dynamique peut même être considérée (cf. section 2.4.3 page 59).

### 2.3.2 Facteurs de forme

Avec la discrétisation des surfaces en éléments apparaît l'importante notion des *facteurs de forme*. Un facteur de forme est un nombre associé à une paire d'éléments. Il quantifie leur influence mutuelle : deux éléments proches et se faisant face échangent plus d'énergie que deux éléments distants et de biais.

La notion de facteurs de forme est présente de façon explicite ou implicite dans les algorithmes de résolution de radiosité ; ils sont explicites si on les calcule pour résoudre l'équation, implicites si le comportement de l'algorithme reflète l'existence des facteurs de forme sans pour autant les calculer. Les algorithmes basés sur le lancer de rayons en font généralement une utilisation implicite : les rayons échantillonnés ont d'autant plus de chance de transiter d'un élément à l'autre que ces derniers partagent un facteur de forme élevé.

Plus rigoureusement, la valeur d'un facteur de forme reflète l'angle solide occupé par un patch dans le champ de vision de l'autre. Cette information peut être extraite de l'équation de radiosité : pour chaque couple de patches $(P_i, P_j)$ d'aires $(A_i, A_j)$, le reste des notations étant similaire à l'équation (2.6), on définit les facteurs de forme $F_{ij}$ par les équations ci-dessous.

$$B(x) = B_e(x) + \rho_d(x) \int_{y \in Scene} B(y) \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y) dy$$

On définit

$$B_i = \frac{1}{A_i} \int_{x \in P_i} B(x) dx$$

Ce qui nous permet d'écrire en intégrant

$$\begin{aligned}
A_i B_i &= \int_{x \in Pi} B_e(x) dx + \rho_d(x) \iint_{x \in Pi, y \in Scene} B(y) \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y) dy dx \\
A_i B_i &= A_i B_{e_i} + \rho_{d_i} \iint_{x \in Pi, y \in Scene} B(y) \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y) dy dx \\
B_i &= B_{e_i} + \rho_{d_i} \frac{1}{A_i} \iint_{x \in Pi, y \in Scene} B(y) \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y) dy dx \\
B_i &= B_{e_i} + \rho_{d_i} \sum_j B_j \underbrace{\frac{1}{A_i} \iint_{x \in P_i, y \in P_j} \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y) dy dx}_{F_{ij}}
\end{aligned}$$

Soit

$$B_i = B_{e_i} + \rho_{d_i} \sum_j B_j F_{ij} \tag{2.8}$$

Les facteurs de forme ne dépendent que de la géométrie de la scène, et peuvent être utilisés pour quantifier les transferts d'énergie entre éléments. La définition des facteurs de forme permet d'écrire l'équation de radiosité

sous forme d'une équation matricielle. Soit $(B_i)_{i\in[1..n]}$ le vecteur (inconnu) des radiosités de chacun des $n$ patches, soit $(M_{ij})_{i,j}$ la matrice des facteurs de forme (incluant les facteurs $\rho_i$), on en déduit l'écriture :

$$B = B_e + M.B \tag{2.9}$$

Cela suppose que les facteurs de forme ont été calculés et sont connus. Présentée sous cette forme, l'équation de radiosité peut alors être résolue par des techniques de calcul matriciel. C'est l'objet de la section 2.3.3 page 55.

#### 2.3.2.1 Calcul analytique des facteurs de forme

La formulation exacte des facteurs de forme de l'équation (2.8) ne permet généralement de les calculer analytiquement que dans les cas simples sans occlusion. De nombreux exemples sont connus [How82], mais couvrent difficilement les cas rencontrés en synthèse d'images. On doit alors déployer des techniques de calcul approché pour évaluer la double intégrale. La difficulté vient de la présence du terme $V(x, y)$.

Il existe deux techniques bien connues pour évaluer les facteurs de forme : l'échantillonnage avec lancer de rayons, et la technique de l'hémicube.

#### 2.3.2.2 Échantillonnage par lignes locales et globales

À $i$ fixé, pour évaluer les facteurs de forme de tous les couples $(P_i, P_j)$, on peut échantillonner des rayons partant de $P_i$, déterminer pour chacun le premier $P_j$ rencontré, et mettre à jour le $F_{ij}$ associé. Il s'agit de l'échantillonnage par *lignes locales*, car pour chaque rayon tracé depuis un élément donné, les calculs sont relatifs à cet élément.

Les rayons peuvent également être lancés sans avoir pour origine un élément en particulier. Dans ce cas, on leur fait traverser la scène de part en part. La probabilité de traverser les surfaces dépend là encore de la taille et de l'orientation de ces dernières. Cela peut servir à estimer les facteurs de forme. Il s'agit de l'échantillonnage par *lignes globales*.

De tels échantillonnages ont été étudiés par Sbert[Sbe93].

#### 2.3.2.3 Hémicube

L'hémicube permet de calculer les facteurs de forme par une technique de projection. Rappelons la forme initiale de l'équation de radiosité (2.6).

$$B(x) = B_e(x) + \rho_d(x) \int_{y\in\text{scène}} B(y)\frac{\cos\theta\cos\theta'}{\pi\ r^2}V(x,y)dy \tag{2.10}$$

En considérant que la radiosité est constante sur chaque élément, on peut scinder l'intégrale sur la scène en une somme d'intégrales sur les éléments

(équation 2.11).

$$B(x) = B_e(x) + \rho_d(x) \sum_{j=1}^{N} B_j \underbrace{\int_{y \in P_j} \frac{\cos\theta \cos\theta'}{\pi \ r^2} V(x,y) dy}_{\text{facteur de forme point-élément}} \tag{2.11}$$

Cela fait apparaître un facteur de forme non pas entre deux éléments, mais entre un point $x$ et un élément $P_j$. Au facteur $\frac{\cos\theta}{\pi}$ près, il s'agit simplement de l'angle solide occupé par $P_j$ sur une sphère centrée en $x$. Le facteur $\cos\theta$ implique une projection sur le plan contenant $x$ (et orthogonal à la normale en $x$). De cette façon, le facteur de forme diminue si $P_j$ est vu à l'horizon plutôt qu'au zénith. Le facteur $\pi$ est une normalisation.

Calculer cette projection d'angle solide pour déduire le facteur de forme est connu comme étant l'*analogie de Nusselt*. Plusieurs algorithmes, comme celui de l'*hémicube*, se basent sur cette analogie.

Plutôt que d'utiliser une (demi-)sphère englobante pour calculer les angles solides, il est possible d'utiliser une moitié de cube, chaque face étant subdivisée en cellules carrées (Figure 2.4 page suivante). La projection d'un élément sur l'hémicube est grossièrement représentée par l'ensemble des cellules qui intersectent cette projection. Le facteur de forme d'un élément est donc approximé par la somme des facteurs de forme de chaque cellule concernée.

La technique de l'hémicube pour calculer les facteurs de forme est très populaire, car elle peut être optimisée facilement sur des cartes vidéo. En effet, les cellules sont des morceaux de plan dont le facteur de forme peut être approché par une formule très simple (équation 2.12). De plus, chaque face du cube se comporte comme un écran, sur lequel une projection de la scène peut être effectuée par des algorithmes implantés au cœur du matériel. L'algorithme classique du *Z-buffer* peut être utilisé.

Pour une cellule de coordonnées $(x, y, z)$, d'aire $\Delta A$, son facteur de forme $\Delta F$ peut être calculé, différemment pour les cellules de la face supérieure et les cellules des faces latérales.

$$\begin{aligned} \Delta F &= \frac{\Delta A}{\pi(x^2+y^2+1)^2} && \text{facteur de forme d'une cellule supérieure} \\ \Delta F &= \frac{z\Delta A}{\pi(1+y^2+z^2)^2} && \text{facteur de forme d'une cellule latérale.} \end{aligned} \tag{2.12}$$

La technique de l'hémicube est rapide mais relativement imprécise, elle peut générer de nombreux artefacts visibles. Elle autorise cependant à calculer les facteurs de forme à la volée, ce qui peut permettre de résoudre l'équation (2.9) sans maintenir l'intégralité de la matrice $M$ en mémoire. Dans les résolutions par relaxation (section 2.3.3.3 page 56), c'est une économie avantageuse.

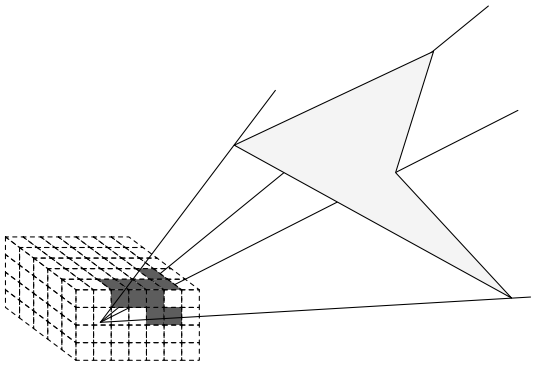


FIG. 2.4 – L'hémicube se comporte comme un écran dont les cellules représentent des zones élémentaires de projection.

### 2.3.3 Résolution

La formulation de l'équation de radiosité sous forme intégrale (2.6) permet de l'exprimer différemment, mais ne suffit pas à la résoudre. De même, la discrétisation de la scène en éléments, et le calcul des facteurs de forme, ne sont pas des techniques de résolution à proprement parler : il faut encore trouver une solution à l'équation discrète.

Pour résoudre l'intégrale, on peut déployer des techniques d'éléments finis ; nous présentons brièvement la méthode de Galerkin et des collocations. Pour l'équation discrétisée faisant apparaître les facteurs de forme (2.9), un algorithme numérique de calcul matriciel peut être utilisé. Cette forme est habituellement résolue par des techniques itératives convergentes (section 2.3.3.3 page suivante), présentées également.

#### 2.3.3.1 Méthode de Galerkin

Les surfaces étant discrétisées, la solution de l'équation de radiosité est une connaissance de la « fonction radiosité » sur chacun des éléments. Cette fonction peut être représentée par combinaisons sur une base de fonction prédéterminées. Cependant, la solution exacte n'est pas forcément représentable dans l'espace des fonctions choisi. Il est donc nécessaire de considérer un critère d'optimalité pour juger de la proximité d'une solution proposée avec la solution réelle. Ce critère peut être basé sur le *résidu* $r(x)$ de l'équation (2.6). Celui-ci correspond à la différence entre la fonction radiosité $\widetilde{B}(x)$ trouvée, et le terme de droite $B_e(x) + \int_y K(x,y)\widetilde{B}(y)dy$. Une solution exacte le verrait nul.

$$r(x) = \widetilde{B}(x) - B_e(x) - \int_{y \in S} K(x,y)\widetilde{B}(y)dy \tag{2.13}$$

En définissant un produit scalaire, la méthode de Galerkin choisit, comme optimalité, l'orthogonalité du résidu avec les fonctions de la base choisie.

La décomposition sur une base de fonctions réduit les inconnues à un jeu de coefficients. Dans ce contexte, l'équation (2.6) s'interprète alors comme une équation matricielle. En nommant $N_i$ les fonctions de la base, $B_i$ les coefficients associés, $< f, g >= \int_x f(x)g(x)dx$ le produit scalaire, et enfin $\epsilon_i =< E, N_i >$, on peut montrer [FC94] que l'équation de radiosité peut s'écrire :

$$MB = \epsilon \tag{2.14}$$

avec

$$M_{ij} =< N_i, N_j > - \int_{x \in S} N_i(x)dx \int_{y \in S} K(x,y)N_j(y)dy \tag{2.15}$$

Cette formulation matricielle est un système linéaire à résoudre avec des techniques standards.

L'équation de radiosité constante par élément (2.9) est un cas dérivé de la méthode de Galerkin, en prenant comme fonctions de base

$$N_j(x) = \begin{cases} 1 \text{ si } x \in P_j \\ 0 \text{ sinon} \end{cases}$$

#### 2.3.3.2 Méthode des collocations

La méthode des collocations est une variation de la méthode de Galerkin. L'optimalité est simplifiée : plutôt que d'exiger l'orthogonalité du résidu avec les fonctions de la base, on peut se contenter de sa nullité en un point particulier de chaque élément. Dans le cas de la radiosité constante par élément, on montre que l'on retombe alors sur l'équation linéaire (2.9) utilisant les facteurs de forme.

Des études de la méthode des collocations dans le cas particulier de l'équation de radiosité ont été menées [ACS00, Han, Han03, Seo02]. Leur principal intérêt n'est pas d'apporter un algorithme plus efficace de résolution, mais d'étudier, pour des cas simples, la façon d'évaluer les intégrales nécessaires à la mise au point du système linéaire, et les taux de convergence obtenus en fonction du maillage choisi.

#### 2.3.3.3 Résolution itérative convergente

L'équation de la forme $B = B_e + MB$ s'écrit plus simplement $(I-M)B = B_e$, de la forme $AX = Y$. Cela est alors équivalent à résoudre un système linéaire. L'inversion de la matrice $A$ est théoriquement possible, mais présente deux inconvénients : l'intégralité de la matrice doit être constamment présent en mémoire, et le calcul n'est pas *progressif* : un arrêt prématuré du calcul ne donne pas forcément de résultat « à peu près correct ».

Dans le cas de la radiosité, d'autres techniques sont déployées pour résoudre ce système. Les méthodes les plus connues sont des techniques itératives convergentes, connues sous le nom de *relaxation de Jacobi*, de *Gauss-Seidel*, ou de *Southwell*. Le plus souvent, quelques itérations suffisent à obtenir un résultat correct.

**La relaxation de Jacobi** repose sur la suite suivante :

$$\begin{cases} X^{(0)} & \text{est initialisé à une valeur quelconque, par exemple } B_e \\ X_i^{(n+1)} = \dfrac{Y_i - \sum\limits_{j \neq i}^{N} A_{ij} X_j^{(n)}}{A_{ii}} \end{cases}$$

La convergence est assurée par une matrice $A$ convenable, à diagonale dominante. Ici, cela traduit le fait que l'énergie est absorbée au fur et à mesure

des itérations. Les facteurs de forme doivent donc être corrects (leur somme ligne par ligne vaut 1) et la réflexion diffuse $\rho_d$ est strictement inférieure à 1.

**La relaxation de Gauss-Seidel** est une variation de la relaxation de Jacobi, qui autorise à effectuer l'opération $X^{(n+1)} = f(X^{(n)})$ ligne par ligne en réutilisant au fur et à mesure les valeurs déjà calculées. Par rapport à la relaxation de Jacobi, on gagne en mémoire un vecteur de stockage intermédiaire, et la convergence asymptotique est meilleure. La matrice $A$ doit pour cela avoir une diagonale *strictement* dominante. C'est effectivement le cas pour la radiosité.

$$\begin{cases} X^{(0)} & \text{est initialisé à une valeur quelconque, par exemple } B_e \\ X_i^{(n+1)} = \dfrac{Y_i - \sum\limits_{0 \leq j < i} A_{ij} X_j^{(n+1)} - \sum\limits_{i < j \leq N}^{N} A_{ij} X_j^{(n)}}{A_{ii}} \end{cases}$$

**La relaxation de Southwell** est une autre variation de la relaxation de Jacobi, mais avec une progression monitorée. Dans le cas de Jacobi ou de Gauss-Seidel, la mise à jour de la ligne $i$ de la solution consiste en une propagation d'énergie depuis tous les éléments vers l'élément $i$. Dans le cas de Southwell, on met plutôt à jour l'intégralité de la solution, en regardant ce qui émane d'un seul élément $j$. Dans les deux cas, seule une partie de la matrice est utilisée (une ligne dans le premier cas, une colonne dans le second). L'intérêt de la relaxation de Southwell est qu'il est possible de choisir intelligemment l'élément émetteur : il suffit de sélectionner à chaque fois celui qui a le plus d'énergie à émettre. Les itérations sont donc monitorées pour une convergence plus rapide.

**Interprétation :** Il est possible de donner une interprétation moins mathématique de ces méthodes pour aider à comprendre leur signification. La technique de Jacobi consiste à mettre à jour un élément à la fois, d'après son environnement. Cela revient à calculer l'énergie qu'il reçoit. On parle de méthode par *gathering*. En revanche, la technique de Southwell revient à modifier à chaque étape, tous les éléments d'après l'émission d'un seul. On parle de méthode par *shooting*.
La méthode *gathering* répartit ainsi l'énergie en suivant le trajet inverse de la lumière (ray-tracing), tandis que la méthode par *shooting* suit le trajet des photons (light-tracing).

## 2.4 Améliorations

Au delà des principes de bases (discrétisation, relaxation), de nombreux raffinements ont été apportés aux algorithmes de résolution de la radiosité, pour améliorer le rapport entre précision et temps de calcul. Ces raffinements peuvent intervenir à tous niveaux, de la discrétisation à des interprétations plus subtiles de l'équation. Quelques principes sont exposés.

### 2.4.1 Subdivision supplémentaire

La radiosité étant souvent traitée comme une constante sur chaque élément, la précision peut être augmentée en utilisant des éléments de plus petite taille, au prix d'un calcul plus coûteux. La matrice des facteurs de forme étant carrée, on peut évaluer que l'augmentation de la complexité est quadratique avec la multiplication du nombre d'éléments.

Plutôt que d'augmenter uniformément la charge de calcul en multipliant les éléments, il est possible d'agir plus finement en utilisant deux niveaux de subdivision. Chaque élément peut être subdivisé en sous-éléments. Utiliser uniquement les sous-éléments reviendrait à utiliser des éléments plus petits. L'idée est au contraire de résoudre les propagations d'énergie comme précédemment, d'élément à élément, mais d'ajouter une deuxième passe de répartition de l'énergie des éléments sur leurs sous-éléments.

Supposons la présence de $N$ éléments et $M$ sous-éléments ($M > N$). La matrice des facteurs de forme reste une matrice $N \times N$ pour une étape de relaxation. Une deuxième matrice de facteurs de forme entre éléments et sous-éléments, et de taille $N \times M$, permet d'ajuster la répartition sur les sous-éléments. Mais en aucun cas on ne calcule de matrice $M \times M$.

### 2.4.2 Subdivision hiérarchique

La subdivision hiérarchique est en quelque sorte une généralisation de la subdivision en sous-éléments présentée précédemment. En la répétant à plusieurs niveaux, on peut avoir plusieurs représentations de la scène à différentes granularités. Chaque échange d'énergie peut alors être calculé plus ou moins grossièrement, en choisissant les niveaux de détail, en fonction de l'impact sur la scène.

Par exemple, la précision attendue pour un échange entre deux petits objets proches est supérieures à celle attendue pour les mêmes objets plus distants. En fonction de tels critères, on peut choisir le niveau de détail pour l'échange considéré. La modification de l'énergie sur un certain niveau peut être répercutée sur les autres niveaux : les niveaux supérieurs agrègent l'énergie des niveaux inférieurs, et les niveaux inférieurs se répartissent l'énergie des niveaux supérieurs [HSA91].

### 2.4.3 Subdivision adaptative

L'un des problèmes de la mise au point d'une bonne discrétisation des surfaces, est qu'elle devrait être non régulière pour être efficace : les zones de fortes variations d'éclairage devraient être dotées d'une plus grande précision (divisions plus fines) que celles qui varient peu. Mais les variations sont inconnues a priori puisqu'elles constituent la solution recherchée. Il peut donc être judicieux de prévoir une modification dynamique des maillages pour pouvoir subdiviser une zone qui semble devenir « sensible ». On parle alors de subdivision adaptive.

Remarquons que le *discontinuity meshing* [ATCF90] est une utilisation avancée de cette technique, exigeant du maillage la capacité à aligner les arêtes de ses éléments sur les frontières des ombres.

### 2.4.4 Notion d'importance

L'équation de radiosité dispose d'une équation duale. Il s'agit de l'équation d'*importance*, qui a exactement la même forme. L'importance traduit l'impact qu'a un élément sur l'image que l'on désire calculer. La quantité d'importance, duale de la radiosité $B_i$, se note $I_i$. Chaque élément dispose également d'une « importance intrinsèque » $R_i$. On peut montrer [FC94] que l'équation d'importance est la suivante :

$$I_i = R_i + \sum_{j=1}^{N} \rho_j F_{ji} I_j \tag{2.16}$$

Cette équation est similaire dans sa forme à l'équation de radiosité (2.6), donc sa résolution n'est pas plus aisée. En revanche, l'utilisation simultanée des deux équations peut permettre de minimiser les calculs en détectant les zones à la fois importantes et lumineuses. Dans le cas d'une scène dont seule une faible partie est visible, une économie peut alors être faite des calculs à effectuer sur des zones n'ayant que peu d'intérêt pour l'image [PM95, SAS92]. On retrouve cette notion dans le bi-directional ray-tracing (cf. section 2.2.3 page 47). En économisant du calcul dans les zones non visibles de l'image finale, on crée toutefois une dépendance avec le point de vue. Déplacer la caméra remet alors en cause les calculs.

### 2.4.5 Parallélisation

La parallélisation des calculs est un domaine de recherche de la résolution de l'équation de radiosité [Fun96, GRS95, PC94, SAG94, SW94, YIY97]. Le défi consiste à segmenter le problème en flots parallélisables. L'utilisation de mémoire partagée peut faciliter la tâche [BIS98, BMP93, PRR98, RAPP97, SH00], car il est difficile, au vu des dépendances mutuelles entre les éléments de la scène, de scinder le problème à résoudre en sous-problèmes plus petits.

Deux grandes approches sont envisageables :
- copier intégralement la scène sur chaque machine et regrouper régulièrement les calculs faits indépendamment ; cela limite la taille de la scène à la mémoire de la plus petite machine ;
- distribuer la scène sur les machines ; cela nécessite de faire transiter des données ou des instructions pour compléter un calcul qui ne peut être effectué sur une seule machine.

Les recherches effectuées favorisent la deuxième approche, dont il faut minimiser les communications.

Nous présentons en section 6.2 page 139 les recherches effectuées sur la parallélisation de notre méthode de radiosité (section 4 page 85).

### 2.4.6 Résumé des techniques déployées

À titre informatif, la Figure 2.4.6 page ci-contre tente de regrouper et de classer les différentes techniques rencontrées dans la résolution de l'illumination globale ou de la radiosité.

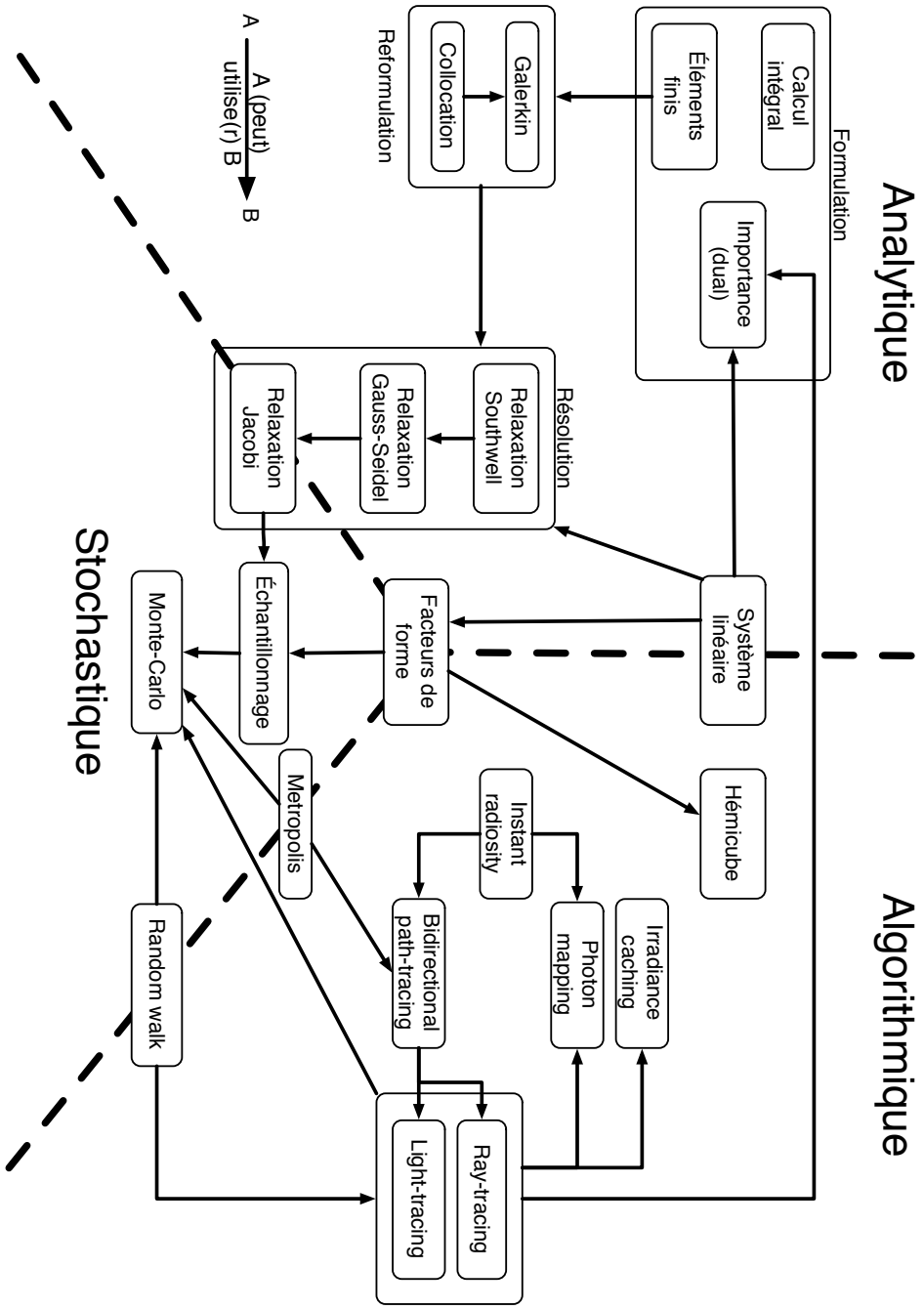



FIG. 2.5 – Tentative d'illustration des méthodes, algorithmes et outils pour résoudre l'équation d'illumination globale. La plupart des méthodes sont en fait hybrides.

# Chapitre 3

# Géométrie discrète

Les données informatiques et les capacités de traitement des processeurs ne se prêtent pas toujours à une transcription directe des mathématiques théoriques sous forme d'algorithmes de traitement. L'instabilité numérique est un exemple d'écueil souvent rencontré lors de l'implantation d'un algorithme théoriquement exact. C'est une première sensibilisation à l'imperfection des traitements numériques informatisés. Mais la nature même des données pousse elle aussi à s'éloigner des mathématiques traditionnelles pour s'adapter aux contraintes technologiques sous-jacentes. Toutes les valeurs de $\mathbb{R}$ n'étant pas accessibles à un processeur, forcément limité par la taille de ses registres, il convient de se munir algorithmiquement de structures permettant de représenter des domaines dans lesquels les calculs sont exacts.

Un algorithme ne traite généralement pas un signal analogique mais un signal numérique, qui est par nature échantillonné. Une transformation de ce signal, ou la génération de données, doit être conforme à l'espace d'arrivée. Prenons par exemple une image sous forme d'un ensemble de pixels carrés : une rotation de l'image est quelque chose de difficile à réaliser, car l'application mathématique de la rotation ne génère pas forcément des coordonnées d'arrivée entières. Un simple arrondi ne peut être appliqué pour pallier cela car il risque :

- d'apporter des déformations,
- de créer des trous dans l'image en n'atteignant pas l'intégralité de l'espace d'arrivée,
- de perdre de l'information en superposant des pixels.

Ce simple exemple suffit à illustrer la difficulté du passage du continu au discret. Les traitements numériques nécessitent un encadrement théorique particulier pour avoir un sens dans un espace discret.

La *Géométrie Discrète* est une discipline qui s'intéresse à la représentation d'objets dans un espace discret. Cela concerne à la fois leur construction, leurs propriétés, et leurs relations avec le continu. La plupart des recherches sont effectuées en 2D et en 3D (droites, plans...), certains résultats sont

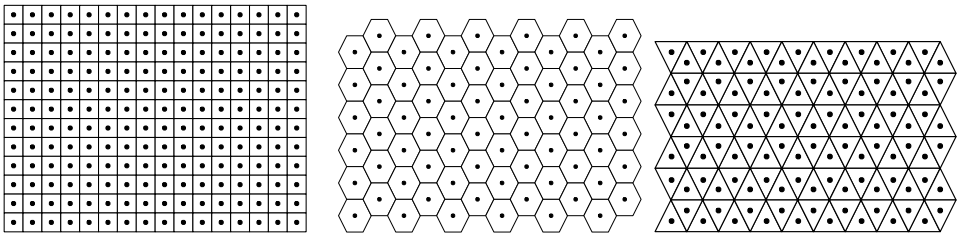
généralisés en $n$D.

Le présent chapitre est une introduction aux notions de la géométrie discrète qui nous ont été utiles dans le cadre de la radiosité discrète. Il introduit les concepts fondamentaux, les problématiques associées, et les travaux effectués sur ces différents aspects. Inspirée de l'excellente présentation effectuée par Sivignon [Siv04], notre introduction n'en est pas moins qu'un sous-ensemble limité à ce dont nous avons eu l'usage.

## 3.1 Notions de géométrie discrète

### 3.1.1 Espace discret : pixel et voxel

On nomme espace discret en dimension $n$ un pavage régulier de $\mathbb{R}^n$. Si l'on se limite à des polygones réguliers, le nombre de possibilités est restreint : en 2D, trois pavés peuvent convenir (carré, hexagone, triangle équilatéral), et un seul en 3D (le cube). Le pavé est positionné par les coordonnées de son centre de gravité, qui peuvent être exprimées dans un repère discret tel $\mathbb{Z}^n$. En 2D, c'est le pavé carré qui est le plus utilisé (Figure 3.1) car il est le plus facile à manipuler. Le pavage en cellules hexagonales est assez courant dans d'autres domaines car il a la propriété intéressante qu'une cellule est à la même distance de ses six voisins immédiats, ce qui n'est pas le cas des cellules carrées et de leurs huit voisins immédiats.



(a) (b) (c)

FIG. 3.1 – (a) Pavage carré. (b) Pavage hexagonal. (c) Pavage triangulaire

Un pavé de l'espace 2D est appelé *pixel* (pour *picture element*). Son pendant tri-dimensionnel est appelé *voxel* (pour *volume element*). Leur assemblage permet de représenter des objets (Figure 3.2 page suivante).

Manipuler les pixels ou les voxels passe par la compréhension des lois qui régissent l'espace discret, très différentes des lois valables dans le continu. La notion de droite en est un exemple typique. La figure 3.3 page suivante montre deux objets discrets (ensembles de pixels) ressemblant à une droite continue. Peut-on dire que l'un est plus ressemblant que l'autre ? La réponse est ici non, les deux objets sont des droites discrètes d'après des définitions différentes (cf. section 3.2.2 page 73). L'un des premiers problèmes en géométrie discrète est de trouver de nouvelles définitions de primitives. Il peut y avoir plusieurs définitions, équivalentes ou non.

### 3.1.2 Voisinage, connexité, séparabilité

Pour décrire la proximité de deux points, la notion de *voisinage* dans un espace discret est assez intuitive. Elle se formalise topologiquement, et permet de définir les notion de connexité et de séparabilité.

Pour des cellules carrées, on définit en 2D deux voisinages différents, le 4-voisinage et le 8-voisinage, ainsi nommés en fonction du nombre de voisins

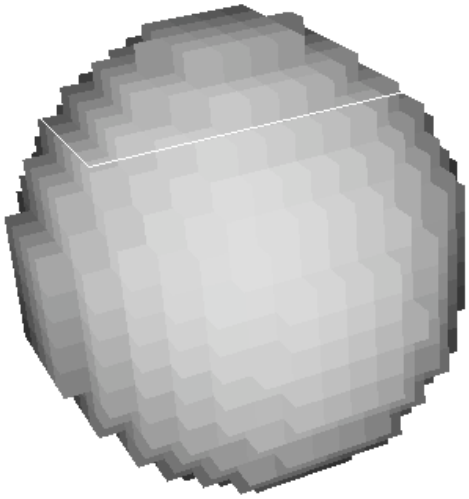
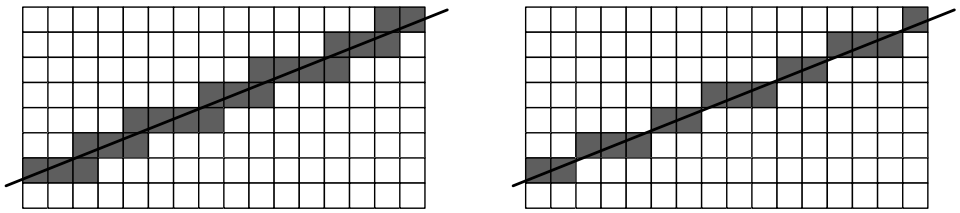


FIG. 3.2 – Un objet constitué de voxels



(a) (b)

FIG. 3.3 – (a) et (b) : deux objets discrets ressemblant à une droite.

acceptés ; en 3D, on est amenés à définir trois voisinages différents, 6, 18 ou 26-voisinage (Figure 3.4 page suivante).

Le voisinage a une origine topologique, traduisant la dimension des parties communes aux éléments. En 3D, cette partie commune peut être un sommet (dimension 0), une arête (dimension 1), ou une face (dimension 2). On parle alors de façon équivalente de 0-voisinage, 1-voisinage ou 2-voisinage.

Le voisinage a également une caractérisation métrique, associée à une distance. Soit deux pixels $(x_a, y_a)$ et $(x_b, y_b)$ ; on définit la *distance de Manhattan* $d_1(a,b) = |x_a - x_b| + |y_a - y_b|$ et la *distance de l'échiquier* $d_\infty(a,b) = max(|x_a - x_b|, |y_a - y_b|)$. Ces distances s'étendent en $n$D.

Deux éléments $n$-voisins sont aussi dits $n$-adjacents.

**Définition 1 : $k$-chemin**
Un $k$-chemin est une suite de points discrets $\{p_i\}_{i=0...n}$ tels que $\forall i \in [1..n]$, $p_i$ est $k$-voisin de $p_{i-1}$.

**Définition 2 : objet $k$-connexe**
Un objet discret X est dit $k$-connexe s'il existe pour tout couple de points

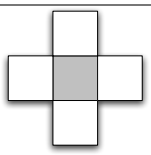
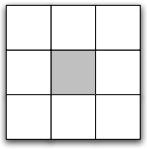
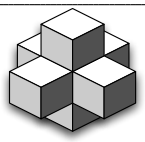
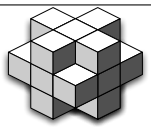
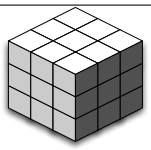
| illustration | nom de dénombrement | nom topologique | caractérisation en distance |
|---|---|---|---|
|  | 4-voisinage | 1-voisinage | $d_1(a,b) = 1$ |
|  | 8-voisinage | 0-voisinage | $d_\infty(a,b) = 1$ |
|  | 6-voisinage | 2-voisinage | $d_1(a,b) = 1$ |
|  | 18-voisinage | 1-voisinage | $d_1(a,b) \leq 2$ et $d_\infty(a,b) = 1$ |
|  | 26-voisinage | 0-voisinage | $d_\infty(a,b) = 1$ |

FIG. 3.4 – Voisinages en dimensions 2 et 3

de $X$ un $k$-chemin qui les relie.

**Définition 3 : $k$-composante connexe**
Une $k$-composante connexe est un ensemble maximal $k$-connexe.

**Définition 4 : séparation**
Une composante $k$-séparative est un ensemble dont le complémentaire est constitué d'exactement deux composantes $k$-connexes.

**Définition 5 : arc et courbe**
Un arc $k$-connexe est un $k$-chemin où chaque point n'a que deux voisins, aux deux extrémités près qui peuvent n'en avoir qu'un. Si l'arc est fermé et n'a pas d'extrémités, on parle de courbe $k$-connexe.

### 3.1.3 Courbes et surfaces

La courbe a pu être définie aisément à partir d'un $k$-chemin. Mais dans un domaine discret, elle ne répond pas forcément automatiquement aux propriétés classiques attendues. L'observation du théorème de Jordan en est une illustration.

#### 3.1.3.1 Théorème de Jordan

Le *théorème de Jordan* stipule qu'une courbe simple, fermée, dans un sous-ensemble connexe de $\mathbb{R}^2$, sépare l'espace en deux composantes connexes. En discret, il faut rajouter des informations sur le type de connexité. La figure 3.5 illustre ce problème : la courbe de gauche, 8-connexe, est complémentaire d'une seule composante 8 connexe, qui peut être vue comme deux composantes 4-connexes. De même, la courbe de droite, 4-connexe, est complémentaire de trois composantes 4-connexes ou deux composantes 8-connexes. Les connexités 4 et 8 sont donc complémentaires du point de vue topologique. En 3D, 6 et 26, 6 et 18, sont également complémentaires.

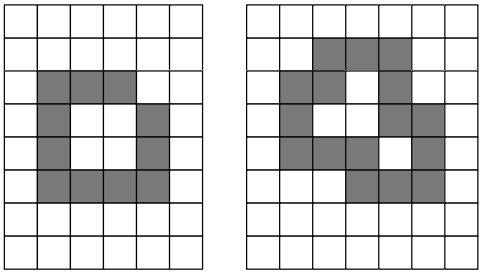


FIG. 3.5 – Le théorème de Jordan ne se transpose pas directement en discret. La courbe de gauche, 8-connexe, est complémentaire d'une seule composante 8 connexe, qui peut être vue comme deux composantes 4-connexes. De même, la courbe de droite, 4-connexe, est complémentaire de trois composantes 4-connexes ou deux composantes 8-connexes.

#### 3.1.3.2 Surfaces

Si la définition de la courbe reste assez simple, la surface est une notion plus difficile à développer dans un contexte discret. En réalité, il existe plusieurs définitions non équivalentes des surfaces en discret. Si l'on nomme *spel* un élément de l'espace discret (pixel en 2D, voxel en 3D...), on peut remarquer qu'il est composé de cellules de dimensions inférieures (faces, arêtes, sommets). Une surface peut alors être définie de plusieurs façons selon l'intérêt qu'on porte à cette subdivision.

La première approche, dite *homogène*, donne une vision unifiée des surfaces et des objets : ils sont tous deux un assemblage de spels, de même

nature. La deuxième approche, dite *hétérogène* voit les surfaces comme un ensemble de cellules de dimensions inférieures.

#### 3.1.3.3 Approche homogène

**Définition 1** (dimension $n$) : [USH82]
Un spel appartient au bord de l'objet si un de ses voisins (au sens de la connexité du complémentaire de l'objet) n'appartient pas à l'objet.

Cette définition est celle d'une frontière plutôt que d'une surface. Elle se rattache au théorème de Jordan si l'on choisit les couples de connexité adaptés.

**Définition 2** (dimension 3) : [Mal97, MR81]
$S \subset \mathbb{Z}^3$ est une surface si et seulement si $S$ sépare $\mathbb{Z}^3$ en deux composantes 6-connexes, et tout voxel de $S$ est 6-adjacent à chaque composante de $\mathbb{Z}^3 \setminus S$.

Cette définition pose cependant problème [Mal96]. La figure 3.6 est une sorte de « tore pincé ». C'est une surface au sens de la définition précédente, mais il est difficile d'admettre qu'il s'agisse d'une « bonne » surface. En continu, le voisinage du point de pincement ne serait pas homéomorphe à un disque (Figure 3.7).

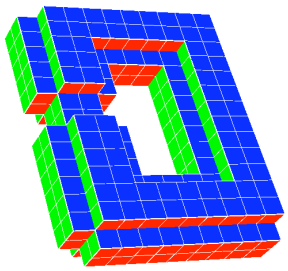


FIG. 3.6 – Ce « tore pincé » pose problème. Il est difficile d'admettre qu'il corresponde à une surface discrète.

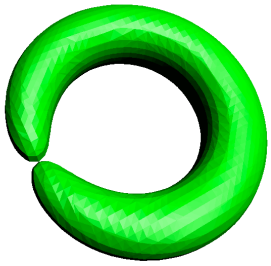


FIG. 3.7 – En continu, ce « tore pincé » est pathologique, car le point de pincement n'a pas un voisinage homéomorphe à un disque.

**Définition 3** (dimension $n$) : « Simplicity surfaces ». Dans cette définition plus récente [CB98], une surface est caractérisée localement par le voisinage d'un point. Le voisinage d'un point doit être une courbe fermée, selon une certaine relation faisant appel à un graphe particulier. Ce graphe est construit sur des informations homotopiques, selon que les points sont des points simples ou non.

Cette définition des surfaces permet de généraliser les notions de 6- et 26-surfaces de Morgenthaler [MR81], ainsi que la notion de 26-surface

forte [MB99].

Perroton soulève [Per94] que la représentation homogène a des inconvénients par rapport à ce qu'on peut attendre d'une surface :

- la surface d'un objet diffère de la surface de son complémentaire ;
- les deux surfaces que l'on aimerait obtenir (intérieure et extérieure) pour un objet avec une cavité, peuvent être confondues si l'objet n'est pas assez épais.

#### 3.1.3.4 Approche hétérogène

**Définition 1** (dimension 3) :

- un **surfel** $s$ est la face commune à toute paire $\{v, \bar{v}\}$ de voxels 6-adjacents. On peut noter $s = \{v, \bar{v}\}$
- on appelle **surface** d'un objet discret 3D $O$, l'ensemble des surfels $s = \{v, \bar{v}\}$ tels que $v \in O$ et $\bar{v} \notin O$.

Cette définition peut être généralisée en dimension $n$ [Her92]

### 3.1.4 Objets discrets : discrétisation et reconnaissance

Une fois défini l'espace discret, on peut y placer des objets, lesquels sont représentés par un ensemble de spels. Un tel ensemble peut avoir deux origines : soit il a été calculé à partir d'un objet continu connu, et est censé le représenter, soit il a été fourni directement, par acquisition ou par définition, mais n'a *a priori* pas de signification intrinsèque. Cela correspond à deux problématiques. Dans le premier cas, c'est un processus de *discrétisation* qui a transformé un objet continu en objet discret. La mise au point d'un tel algorithme dépend des propriétés exigées de la forme discrète. Dans le deuxième cas, une phase de *reconnaissance* peut être envisagée pour donner un sens aux données.

#### 3.1.4.1 Discrétisation

Si un objet discret peut être vu comme un ensemble de spels, il peut également être interprété comme un sous-ensemble de spels de l'espace complet. Dans ce cas, il est représenté par une fonction binaire répondant *appartient/n'appartient pas* pour tout spel. La discrétisation revêt alors un rôle d'échantillonnage.

On attend généralement d'un objet discrétisé qu'il conserve les propriétés topologique de l'objet continu correspondant : connexité, séparativité... L'algorithme de discrétisation doit donc tenir compte de ces propriétés pour calculer la meilleure représentation discrète possible. Les algorithmes de discrétisation sont présentés en section 3.3 page 80.

Pour assurer une reconstruction de l'objet originel, l'information binaire est souvent insuffisante. S'il est important de pouvoir faire cette reconstruction, un spel peut être associé non plus à la valeur 0 ou 1, mais à tout nombre réel dans $[0; 1]$. Cela permet de quantifier la proportion d'objet et de fond qui se trouvent dans un spel, sur le bord d'un objet (cf. section 3.3.2 page 81). En logique floue, cette valuation permet aussi de traiter les cas de contours imprécis.

#### 3.1.4.2 Reconnaissance

Les appareils de mesure fournissent souvent des données échantillonnées comme résultats de leurs acquisitions. Elles peuvent être interprétées par un superviseur qui connaît la nature des objets observés. Cependant, pour structurer l'information obtenue, une phase de reconnaissance peut être effectuée. Celle-ci consiste à déterminer si un ensemble de voxels donné est compatible avec la discrétisation d'un certain objet continu. On peut ainsi reconnaître une droite, un plan, même incomplets, dans un ensemble non structuré. Il est alors possible d'utiliser des algorithmes de traitement adéquats, pour la visualisation par exemple. Les algorithmes de reconnaissance sont abordés sans être détaillés en section 3.2.6 page 79.

## 3.2 Mise en œuvre de la géométrie discrète

### 3.2.1 La grille de discrétisation

À l'espace discret est associé un repère discret représentant les spels. Si la visualisation privilégie les spels, la discrétisation s'appréhende mieux sur la grille des centres. Il ne s'agit que d'une seule et même chose, mais il importe de ne pas confondre le *bord d'un spel* avec une *arête de la grille*, définie entre deux centres de spels voisins.

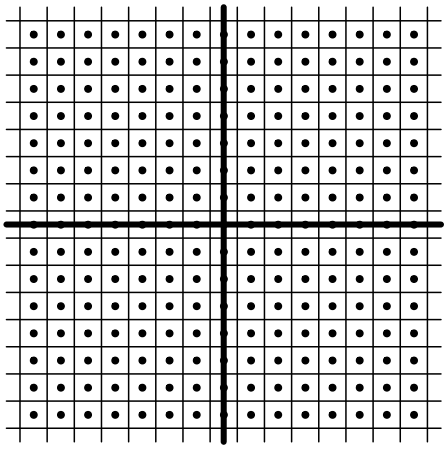


FIG. 3.8 – Représentation des spels et de leur centres

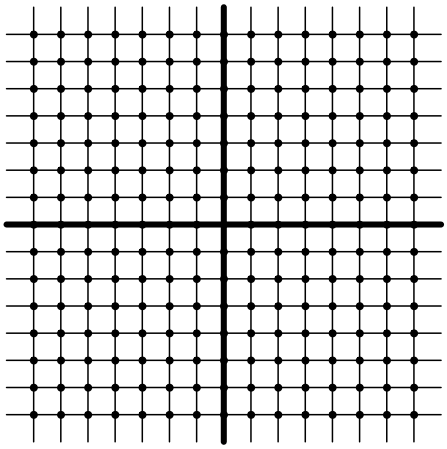


FIG. 3.9 – Représentation de la grille des centres

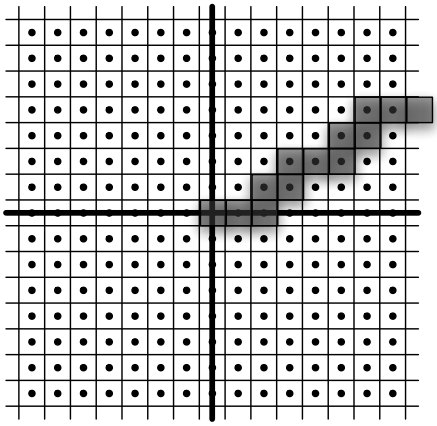


FIG. 3.10 – Représentation d'un objet dans les spels

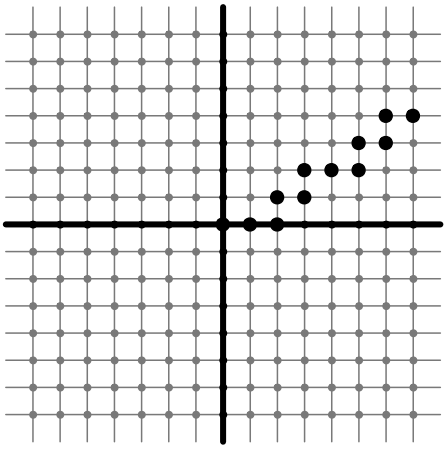


FIG. 3.11 – Représentation d'un objet sur la grille

### 3.2.2 Droites discrètes

La droite discrète est une primitive intéressante pour introduire la notion d'arithmétique en géométrie discrète. Considérons la droite continue d'équation $3x - 7y = 0$. Cette droite continue passe par quelques points entiers, mais la plupart du temps elle intersecte des arêtes de la grille (Figure 3.12). Le choix des points entiers voisins de ces intersections correspond à la définition de la droite discrète.

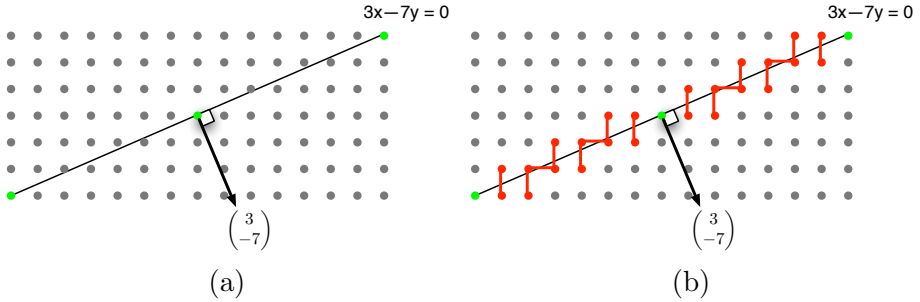



FIG. 3.12 – Une droite continue (a) peut passer par certains points de la grille et intersecter de nombreuses arêtes (b)

#### 3.2.2.1 Définition algorithmique

En première approche, on peut donner une définition *algorithmique* d'une discrétisation. Notons d'abord que le vecteur normal de la droite lui donne une orientation. On peut donc définir un intérieur et un extérieur, ce qui serait notamment utile s'il s'agissait en fait d'un segment de droite bordant un objet. Plusieurs choix sont alors possibles :

- discrétisation OBQ (Object Boundary Quantization) : pour une intersection avec une arête, on choisit le point à l'intérieur de l'objet (Figure 3.13 page suivante).
- discrétisation BBQ (Background Boundary Quantization) : pour une intersection avec une arête, on choisit le point à l'extérieur de l'objet (Figure 3.14 page suivante).
- discrétisation GIQ (Grid Intersect Quantization) : pour une intersection avec une arête, on choisit le point le plus proche de l'intersection (Figure 3.15 page suivante).
- discrétisation par supercouverture : tout spel intersectant la droite fait partie de sa discrétisation (Figure 3.16 page suivante).

Ces différentes discrétisations ne sont pas équivalentes. Les discrétisations OBQ et BBQ nécessitent une orientation. La supercouverture fait apparaître des « bulles » lorsque la droite passe par un sommet de spel.

L'arithmétique en géométrie discrète permet de donner une définition *mathématique* plutôt qu'algorithmique à ces différentes possibilités.

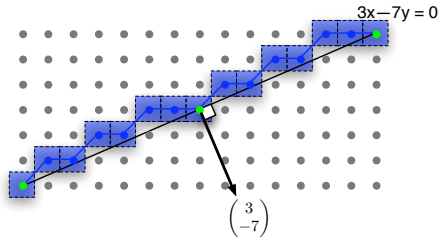



FIG. 3.13 – La discrétisation OBQ

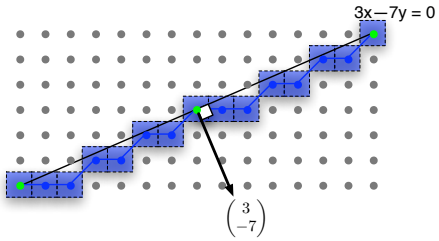



FIG. 3.14 – La discrétisation BBQ

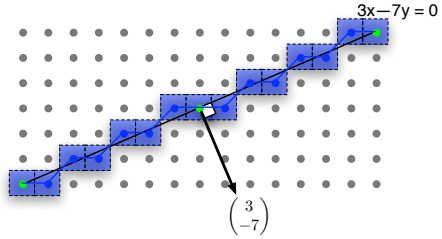



FIG. 3.15 – La discrétisation GIQ

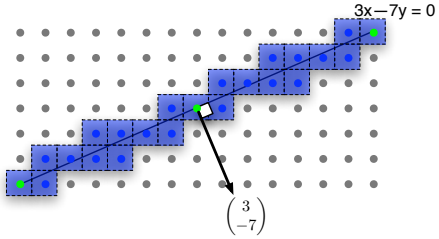



FIG. 3.16 – La discrétisation supercouverture

#### 3.2.2.2 Définition arithmétique

Reveillès a proposé en 1991 [Rev91] la définition fondamentale de la droite discrète 2D, notée $D(a, b, \mu, \omega)$

$$D(a, b, \mu, \omega) = \{(x, y) \in \mathbb{Z}^2 \quad / \quad 0 \leq ax - by + \mu < \omega\} \tag{3.1}$$

Cette définition peut s'expliquer ainsi : à chaque point discret $(x_d, y_d)$ de la grille qui n'est pas exactement sur la droite d'équation $ax - by + c = 0$, on peut associer l'erreur $r = ax_d - by_d + \mu$, appelé le *reste*. $\mu$ peut être choisi différent de $c$.
Si ce reste est dans un certain intervalle, on peut considérer que le point appartient à la droite discrète. Les paramètres $(a, b)$ donnent l'orientation de la droite, $\mu$ permet de la décaler, et $\omega$ contrôle l'épaisseur de la bande. Un choix judicieux des paramètres $\mu$ et $\omega$ permet de retrouver les discrétisations présentées précédemment (cf. section 3.2.2.1 page précédente).

Soit la droite continue d'équation $ax - by + \mu = 0$

- OBQ correspond à $D(a, b, \mu, \max(|a|, |b|))$ [Vit99, Rev91]
- BBQ correspond à $D(a, b, \mu + \max(|a|, |b|) - 1, \max(|a|, |b|))$ [Vit99, Rev91]
- GIQ correspond à $D(a, b, \mu + \lfloor \frac{b}{2} \rfloor, \max(|a|, |b|))$ [And00, LW00]
- la supercouverture correspond à $D(a, b, \mu + \frac{|a|+|b|}{2}, |a| + |b| + 1)$ [And00, LW00]

Il apparaît assez clairement que les droites OBQ, BBQ et GIQ ont la même épaisseur mais sont légèrement décalées, tandis que la supercouverture est

plus épaisse.

En contrôlant l'épaisseur de la droite, le paramètre $\omega$ en caractérise aussi la connexité. Reveillès a établi que

- si $\omega < \max(|a|, |b|)$, alors $D$ n'est pas 8-connexe (Figure 3.17) ;
- si $\omega = \max(|a|, |b|)$, alors $D$ est 8-connexe d'épaisseur minimale (Figure 3.18) ;
- si $\max(|a|, |b|) < \omega \leq |a| + |b|$, alors $D$ est *-connexe (8-connexe mais non 4-connexe d'épaisseur minimale) (Figure 3.19) ;
- si $\omega = |a| + |b|$, alors $D$ est 4-connexe d'épaisseur minimale (Figure 3.20) ;
- si $\omega > |a| + |b|$, alors $D$ est « épaisse » (Figure 3.21 page suivante).

Une droite discrète est dite **naïve** ($\omega = \max(|a|, |b|)$) si elle est d'épaisseur minimale pour être 8-connexe, et **standard** ($\omega = |a| + |b|$) si elle est d'épaisseur minimale pour être 4-connexe. Dans les illustrations de ces connexités, le paramètre $\mu$ a été fixé à 0. Le voxel $(0, 0)$ est au pied du vecteur normal.

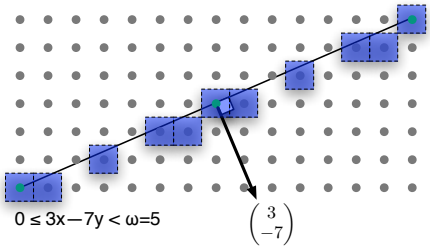



FIG. 3.17 – $\omega < \max(|a|, |b|)$, la droite discrète est déconnectée

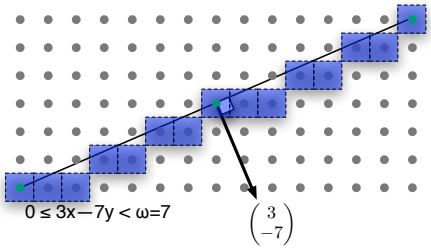



FIG. 3.18 – $\omega = \max(|a|, |b|)$, la droite discrète est 8-connexe et dite naïve

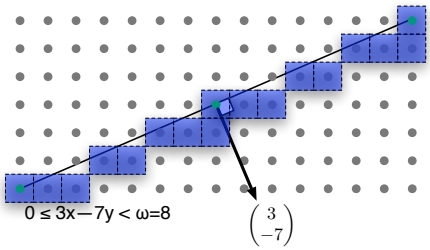



FIG. 3.19 – $\max(|a|, |b|) < \omega < |a| + |b|$, la droite discrète est *-connexe

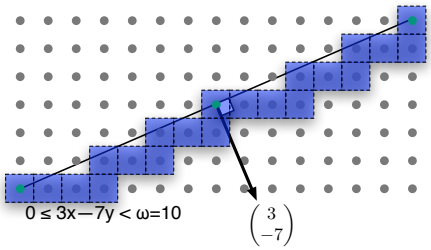



FIG. 3.20 – $\omega = |a| + |b|$, la droite discrète est 4-connexe et dite standard

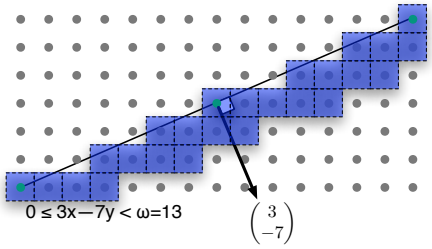



FIG. 3.21 – $\omega > |a| + |b|$,
la droite discrète est épaisse

### 3.2.3 Droites discrètes 3D

À l'instar des droites discrètes 2D, les droites discrètes 3D ont plusieurs définitions. Arithmétiquement, cela reste toutefois proche de la définition de Reveillès.

Isabelle Debled-Rennesson a proposé [DR95] une formulation de la droite 3D, qui peut se lire comme l'intersection de deux plans discrets particuliers ; ces plans discrets étant simplement le plongement et l'extrusion de deux droites discrètes 2D dans l'espace. Les plans sont définis formellement en section 3.2.4 page suivante.
Cette droite 3D discrète est notée $D(a, b, c, \mu, \mu', \omega, \omega')$. On se place dans le cas $a \geq b \geq c > 0$, les autres cas pouvant être déduits par symétrie.

$$D(a,b,c,\mu,\mu',\omega,\omega') = \left\{(x,y,z) \in Z^3 \;/\; \begin{cases} 0 \leq cx - az + \mu < \omega \\ 0 \leq bx - ay + \mu' < \omega' \end{cases} \right\} \quad (3.2)$$

À l'inverse de l'interprétation comme intersection de deux plans, cela revient à dire que la droite 3D discrète se projette en droites 2D discrètes sur $(Oxy)$ et $(Oxz)$. La troisième projection n'est pas forcément une droite 2D discrète.

Les paramètres $\omega$ et $\omega'$ ont toujours une influence directe sur la connexité [DR95] :

- si $\omega < a$ ou $\omega' < a$, D n'est pas connexe
- si $a \leq \omega \leq a + c$ et $a \leq \omega' \leq a + b$, D est 26-connexe
- si $a + c \leq \omega$ et $a \leq \omega' < a + b$, ou $a + b \leq \omega'$ et $a \leq \omega < a + c$, D est 18-connexe
- si $a + c \leq \omega$ et $a + b \leq \omega'$, D est 6-connexe

On peut remarquer que la droite 3D obtenue dans le cas 6-connexe n'est pas un arc 6-connexe au sens de la définition 5 page 67 à cause de la présence de « bulles » (Figure 3.22 page suivante).

Andres a proposé [And00, And03] un cadre général de définition d'objets discrets linéaires en dimension $n$. Un cas particulier est le modèle de droite 3D discrète standard (6-connexe), définie alors à partir de 3 projections. La contrainte supplémentaire de la troisième projection permet de

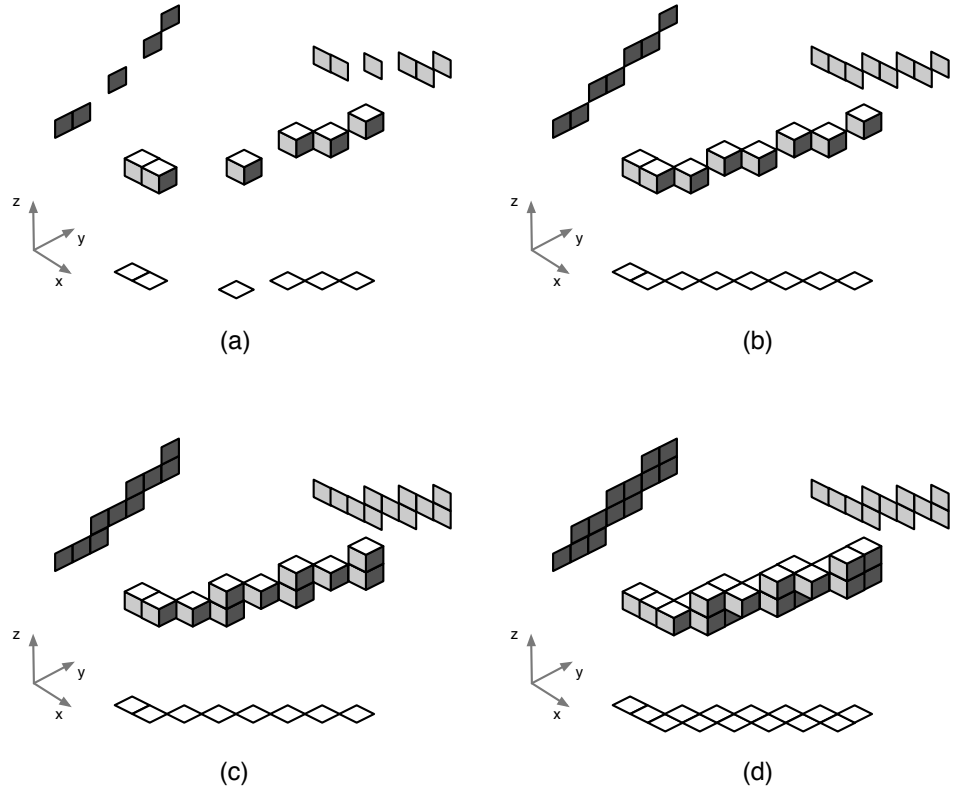



FIG. 3.22 – Droite discrète $D(7, 6, 3, 0, 0, \omega, \omega')$ avec respectivement :
(a) $(\boldsymbol{\omega < 7, \omega'}) = (5, 7)$, la droite est déconnectée
(b) $(\boldsymbol{7 \leq \omega \leq 7 + 3, 7 \leq \omega' \leq 7 + 6}) = (7, 7)$, la droite est 26 connexe
(c) $(\boldsymbol{7 + 3 \leq \omega, 7 \leq \omega' < 7 + 6}) = (10, 7)$, la droite est 18 connexe
(d) $(\boldsymbol{7 + 3 \leq \omega, 6 + 7 \leq \omega'}) = (10, 13)$, la droite est 6-connexe

faire disparaître les bulles.

$$\begin{cases} 0 \leq bx - ay + \mu < \frac{|a|+|b|}{2} & (b > 0 \text{ ou } b = 0 \text{ et } a > 0) \\ 0 \leq cx - az + \mu' < \frac{|a|+|b|}{2} & (c > 0 \text{ ou } c = 0 \text{ et } a > 0) \\ 0 \leq cy - bz + \mu'' < \frac{|a|+|b|}{2} & (c > 0 \text{ ou } c = 0 \text{ et } b > 0) \end{cases} \tag{3.3}$$

### 3.2.4 Plans discrets

À l'image des droites, les plans discrets peuvent être définis de plusieurs façons, comme des discrétisés OBQ, BBQ ou GIQ d'un plan euclidien. De façon assez agréable, on retrouve une équation proche de celle des droites discrètes. On peut définir ainsi le plan de normale $(a, b, c)$, noté $P(a, b, c, \mu, \omega)$ :

$$0 \leq ax + by + cz + \mu < \omega \tag{3.4}$$

Le paramètre $\omega$ jouant sur l'épaisseur est lié à la connexité. La « non-connexité » d'un plan est plus difficile à formaliser que celle d'une droite, car on peut former des trous dans un plan en conservant une connexité globale. La « non-connexité » locale peut en fait être définie par la notion de tunnel, c'est à dire une façon pour le complémentaire de « passer à travers le plan ». Si le tunnel peut être réalisé par un coin, on parle de 0-tunnel ; par une arête, de 1-tunnel ; par une face, de 2-tunnel (Figure 3.23).

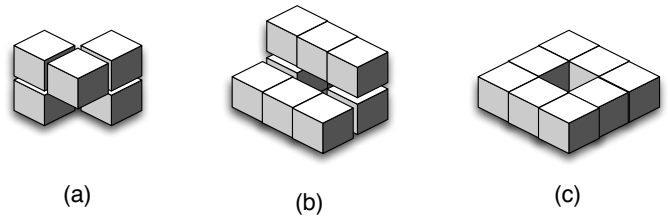



FIG. 3.23 – La non-connexité d'un plan peut être décrite par des tunnels. (a) 0-tunnel, (b) 1-tunnel, (c) 2-tunnel

Les liens entre l'épaisseur et la connexité du plan sont établis dans [AAS97]. On se place dans le cas $0 \leq a \leq b \leq, c \neq 0$

- si $\omega < c$, $P$ a des 2-tunnels
- si $c \leq \omega < b + c$, $P$ a des 1-tunnels
- si $b + c \leq \omega < a + b + c$, $P$ a des 0 tunnels
- si $a + b + c \leq \omega$, $P$ n'a pas de tunnels

- si $\omega < c$, $P$ est 26 connexe ou non connexe.
- si $\omega = c$, $P$ est 18-connexe
- si $c < \omega < b + c$, $P$ est 18-connexe ou 6-connexe
- si $b + c \leq \omega$, $P$ est 6-connexe

Quant aux discrétisations obtenues, on retrouve des résultats très semblables à ceux des droites :

- $P(a, b, c, \mu, \max(|a|, |b|, |c|))$ est un plan naïf OBQ.
- $P(a, b, c, \mu + \max(|a|, |b|, |c|) - 1, \max(|a|, |b|, |c|))$ est un plan naïf BBQ.
- $P(a, b, c, \mu + \lfloor \frac{\max(|a|,|b|,|c|)}{2} \rfloor, \max(|a|, |b|, |c|))$ est un plan naïf GIQ.
- $P(a, b, c, \mu + \frac{|a|+|b|+|c|}{2}, |a| + |b| + |c|)$ est un plan standard GIQ.
- $P(a, b, c, \mu + \lfloor \frac{|a|+|b|+|c|}{2} \rfloor, |a| + |b| + |c| + \delta)$ est la supercouverture du plan continu ($\delta = 1$ si $|a| + |b| + |c|$ est pair, 1 sinon)

### 3.2.5 Autres objets discrets

En plus des droites et des plans, il existe des paramétrisations d'autres primitives telles les cercles, les sphères, les ellipses, les simplexes... Ces travaux ne sont pas détaillés ici [AJ97, And03, And00, AAS97].

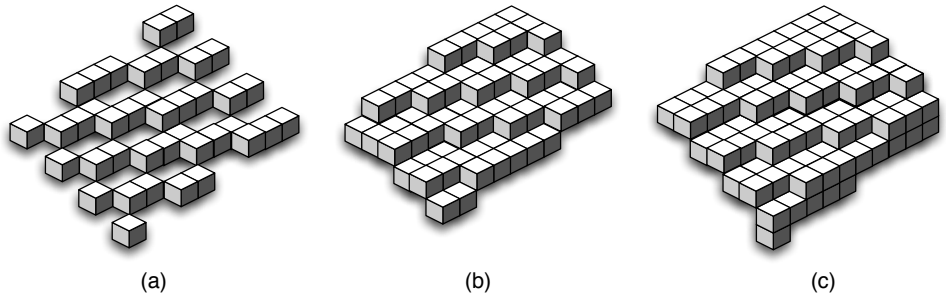



FIG. 3.24 – Plan discret $P(6, 13, 27, 0, \omega)$ avec respectivement :
(a) $\omega = 15 < 27$, $P$ a des 2-tunnels
(b) $\omega = 27 = c$ , $P$ est 18-connexe (naïf)
(c) $\omega = 46 > 13 + 27$, $P$ est 6-connexe (standard)

### 3.2.6 Reconnaissance

Dans les sections précédentes, nous avons mis en évidence le fait que la géométrie peut décrire un objet comme un ensemble de spels. Le problème inverse est un autre thème de recherche : soit un ensemble de spels, peut-il être décrit par un objet discret ?

Cela peut avoir un intérêt pour reconstruire une information incomplète, comme un plan à partir d'un nuage de spels. Cela peut également servir à segmenter un objet complexe en sous-objets plus simples. Les algorithmes de reconnaissance les plus puissants sont dynamiques et peuvent modifier leur réponse au fur et à mesure que l'on ajoute ou enlève des éléments. De nombreux travaux de recherches sont menés sur ces problématiques [DR95, Vit99, Buz03, Siv04].

## 3.3 Algorithmes de discrétisation

La discrétisation d'un objet continu dans un espace discret peut être abordée sous deux angles différents : par génération de l'objet d'après sa description abstraite, ou par tramage de l'espace pour la découvrir. La génération est la réalisation d'un algorithme qui, d'après les paramètres caractérisant l'objet (par exemple $(a, b, c, \mu, \omega)$ pour une droite), produit l'ensemble des voxels correspondants. Le tramage correspond à un échantillonnage de l'espace, qui découvre et assemble les voxels appartenant à l'objet.

### 3.3.1 Génération d'après la description

#### 3.3.1.1 Tracé incrémental

La connaissance d'une équation, comme pour la droite discrète, ne signifie pas pour autant que l'on dispose d'un algorithme efficace de construction. Un tel algorithme devrait permettre de ne générer que les voxels adéquats, et utiliser l'arithmétique entière pour garantir l'exactitude des tests.

L'un des premiers algorithmes incrémentaux est le tracé de la droite 2D de Bresenham, qui n'avait de définition qu'algorithmique, et non arithmétique. En propageant et en mettant à jour une erreur, l'algorithme peut décider ou non d'une transition de palier, produisant ainsi les « marches » au fur et à mesure du tracé effectué pixel par pixel. La droite de Bresenham étant incluse dans la définition des droites discrètes de Reveillès (naïve, GIQ), elle peut être prise en compte dans les algorithmes de tracé proposés par ce dernier [Rev91]. Un algorithme de tracé de droite épaisse a également été mis au point, ce qui diffère de l'aspect « fonctionnel » de l'algorithme de Bresenham, pour lequel une droite discrète dispose d'un et un seul pixel par colonne.

Pour les droites discrètes 3D, Isabelle Debled a introduit un algorithme incrémental similaire [DR95]. De façon plus empirique, Kaufman et Cohen avaient également produit un algorithme de génération de droite [COK97], en adaptant l'algorithme de Bresenham, sans considérations topologiques. De manière plus générale, Kaufman est à l'origine de nombreux algorithmes de génération (cf. section 3.3.1.2), souvent plus calculatoires que topologiquement cohérents.

#### 3.3.1.2 Tramage (*rasterization*)

Le tramage d'objets (rasterization) existe depuis longtemps en 2D dans les algorithmes de rendu d'images, qui doivent afficher une scène dans un buffer video représentant les pixels de l'écran. Ce processus est implanté au cœur des cartes graphiques et on y fait souvent référence comme la *scan-conversion*. Le traitement habituel consiste en effet à balayer la scène ligne

par ligne, et à transformer les segments intersectant les objets en suites de pixels.

Par analogie, une discrétisation vers un espace tri-dimensionnel peut être abordé par des algorithmes de *3D scan-conversion*. Kaufman a produit de nombreux algorithmes [KS87] permettant d'obtenir droites, courbes paramétriques, polygones, polyèdres, cylindres, sphères, cônes. Connaissant *a priori* l'objet à construire, il s'agit d'assembler principalement des segments de droites discrets le long des arêtes pour construire des surfaces. Ces méthodes produisent toutefois des objets dont la topologie n'est pas forcément bien contrôlée, et peut varier entre deux orientations.

Un autre algorithme utilisé par Malgouyres [Mal02] consiste à projeter les surfaces d'objets continus sur les trois plans canoniques $(Oxy)$, $(Oxz)$ et $(Oyz)$, puis à appliquer sur chaque projection une technique de rasterization 2D. Ensuite, en calculant l'intersection des rétro-projections des ensembles obtenus, une surface 6-séparatrice prend forme.

Généralement, dans les cas de courbures raisonnables par rapport à l'échelle des voxels, on obtient des surfaces de Jordan.

La poussée de l'utilisation d'algorithmes implantés dans les cartes graphiques a également trouvé écho dans la voxelisation. Encore assez limité dans la pratique, l'algorithme [ED06] donne une interprétation originale de la grille discrète 3D comme un ensemble de textures superposées. La scène est ainsi découpée en tranches (*slice maps*) permettant d'utiliser des algorithmes 2D classiques pour le tramage de chaque tranche.

#### 3.3.1.3 Tissage (*weaving*)

Les algorithmes de tissage consistent à répéter un motif dont l'accumulation forme l'objet à discrétiser. Il s'agit d'un cas particulier de tramage qui peut améliorer les performances, puisque le motif est répété et non re-généré. Un plan discret peut ainsi être décrit par deux composantes non parallèles, dont l'une est répétée le long de l'autre avec un décalage particulier [LW99].

### 3.3.2 Échantillonnage et filtrage

La géométrie discrète travaillant sur des spels utilise principalement des spels binaires, au sens où soit un spel appartient à un objet, soit non. En logique floue, cette information peut être quantifiée non pas pas par un un nombre binaire mais par un flottant entre 0 et 1.

Sans parler de logique floue, le choix de nombre réels se pose déjà en voyant la voxelisation comme un processus d'échantillonnage plutôt que de discrétisation. Le voxel est alors dit « multivalué » (à l'opposé de binaire) et sa valeur peut être par exemple :

– sa distance, depuis un point particulier (comme le centre) à la surface la plus proche ;

– le résultat d'un « filtrage » de l'objet, c'est-à-dire une convolution locale entre la surface continue et un filtre, caractérisant la proportion d'objet et de fond présent dans le voxel.

Le but de la voxelisation comme échantillonnage est moins d'apporter une structure topologique discrète convenable que de représenter l'objet continu dans un autre domaine, dual, permettant la reconstruction inverse.

Kaufman et Srámek ont étudié les techniques de filtrage et de reconstruction après filtrage. Ils d'abord établi l'intérêt d'un filtre boîte pour obtenir une variation linéaire efficace de la « densité » de l'objet autour de sa frontière [SK98]. Puis, à l'aide de filtres gaussiens et de techniques avancées de reconstruction, ils ont proposé une amélioration de la méthode [SK99]. Des implémentations rapides de ces techniques sont possibles, comme la voxelisation incrémentale d'un triangle [IK00], lequel constitue une primitive de base privilégiée pour les maillages.

# Deuxième partie

# Nouvelle approche de la radiosité

# Chapitre 4

# La radiosité traitée sur des voxels

Qu'il s'agisse de l'utilisation du ray-tracing ou du calcul des facteurs de forme par la technique de l'hémicube, le calcul de radiosité repose sur l'évaluation de la visibilité des éléments entre eux. Le lancer de rayons ou la technique de l'hémicube n'ont d'autre vocation que de déterminer, et parallèlement de quantifier, ce qui fait partie du champ de vision de chaque élément.

Le but de la résolution de l'équation de radiosité sur une scène discrétisée en voxels est d'aborder différemment le calcul de la visibilité. Par le biais de la géométrie discrète, il est possible d'exprimer des relations entre les voxels distants, par du calcul arithmétique sur leurs coordonnées entières. Cette particularité se retrouve également dans la représentation des espaces vides : ils peuvent être représentés de la même façon que les espaces pleins.

La radiosité traitée sur les voxels a été abordée en deux étapes : d'abord *via* un simple lancer de rayons discret, puis améliorée par un algorithme quasi linéaire en utilisant une approche originale.

## 4.1 Discrétisation de l'équation

### 4.1.1 Équation continue adaptée aux voxels

Pour traiter la radiosité sur des voxels, il faut d'abord définir l'équation de radiosité sur ce modèle de subdivision. Il s'agit d'une équation discrète. Rappelons la forme de l'équation de radiosité continue (cf. section 2.1.3.5 page 43) :

$$B(x) = B_e(x) + \rho_d(x) \int_{y \in \text{ Scène}} B(y) \frac{\cos\theta \cos\theta'}{\pi ||x-y||^2} V(x,y) dy \qquad (4.1)$$

La technique de l'hémicube fait disparaître le terme $V(x,y)$ en décomposant l'intégrale en une somme de termes, éventuellement nuls, traduisant la visibilité de tous les couples de patches (éléments surfaciques). Nous pouvons réutiliser cette idée pour une discrétisation sur les voxels.

Supposons que nous disposions d'une représentation de la scène sous forme de voxels. En la superposant à la représentation géométrique continue, on observe que chaque voxel contient un morceau de surface. On peut alors définir une représentation en patches de la façon suivante : chaque morceau de surface délimité par un voxel est un patch. Cela suffit pour utiliser l'équation discrète avec facteurs de forme présentée en section 2.3.2 page 50

$$B_i = B_{e_i} + \rho_{d_i} \sum_j F_{ij} B_j \qquad (4.2)$$

Cela n'apporte rien de plus au problème car aucun usage n'est fait des voxels. En revanche, il est plus pertinent de reprendre l'idée de l'hémicube qui peut servir à calculer les facteurs de forme (cf. section 2.3.2.3 page 52), et qui permet de faire disparaître le terme en $V(x,y)$.

En effet, le terme de visibilité $V(x,y)$ est introduit pour écrire l'intégrale sur l'ensemble de la scène. Mais le terme intégré n'est que rarement non nul. Pour pallier cela, il est possible d'inverser la visibilité et de n'effectuer une somme que sur les éléments visibles depuis $x$ (équation 4.3)

$$B(x) = B_e(x) + \rho_d(x) \int_{y \in \text{ Scène}/V(x,y)\neq 0} B(y) \frac{\cos\theta \cos\theta'}{\pi ||x-y||^2} dy \qquad (4.3)$$

Pour réaliser la condition $y \in$ Scène$/V(x,y) \neq 0$, la technique de l'hémicube réalise un échantillonnage des directions de l'espace, représenté par le nombre de cellules des faces de l'hémicube lui-même, projetant sur chacune le premier élément visible. Ce « premier élément visible » peut être déterminé par projection (technique de z-buffer) ou par lancer de rayons. C'est cette dernière idée que nous pouvons reprendre.

On définit la fonction $V(x, \overrightarrow{\sigma})$ (détaillée en 4.1.2.3 page suivante) qui associe au point $x$ le premier point $y$ rencontré dans la direction $\overrightarrow{\sigma}$. L'équation (4.3) page précédente peut donc être réécrite (4.4).

$$B(x) = B_e(x) + \rho_d(x) \int\limits_{\overrightarrow{\sigma}, y=V(x,\overrightarrow{\sigma})} B(y) \frac{\cos\theta(x,y)\cos\theta(y,x)}{\pi||x-y||^2} dy \qquad (4.4)$$

Pour déterminer l'ensemble des directions $\overrightarrow{\sigma}$, on peut définir $\sigma$ comme le point d'une sphère $S_R(x)$ de centre $x$ et de rayon $R$, $\overrightarrow{\sigma}$ dénotant alors le vecteur $\overrightarrow{x\sigma}$. On peut alors reconnaître dans le terme $\cos\theta(y,x)\frac{dy}{\pi||x-y||^2} = \frac{d\sigma}{\pi R^2}$ un élément d'angle solide.

$$B(x) = B_e(x) + \rho_d(x) \int\limits_{\sigma\in S_R(x), y=V(x,\overrightarrow{\sigma})} B(y)\cos\theta(x,y)\frac{d\sigma}{\pi R^2} \qquad (4.5)$$

### 4.1.2 Formulation de l'équation discrète

#### 4.1.2.1 Équation générale

Afin d'obtenir une forme discrète de l'équation (4.5), il est nécessaire de discrétiser l'ensemble des directions considérées $\sigma \in S_R(x)$. De manière générale, l'ensemble des directions doit être fini, et à chaque direction $\overrightarrow{\sigma}$ doit être associé un facteur $A(\overrightarrow{\sigma})$ caractérisant l'angle solide $\frac{d\sigma}{\pi r^2}$ à cette direction. Nous nommerons celui-ci *facteur de direction*.

Soit $D$ un ensemble de directions, l'équation discrète de radiosité que nous établissons sur un ensemble de voxels représentant une scène est la suivante :

$$B(x) = B_e(x) + \rho_d(x) \sum_{\overrightarrow{\sigma}\in D} B(V(x,\overrightarrow{\sigma}))\cos\theta(x, V(x,\overrightarrow{\sigma}))A(\overrightarrow{\sigma}) \qquad (4.6)$$

#### 4.1.2.2 Exemple d'équation

La définition de l'ensemble $D$ des directions peut s'inspirer des cellules de l'hémicube. Soit $\Sigma$ une surface englobante constituée de voxels, comme une sphère [AJ97] ou un cube. Chaque voxel de la surface dispose de surfels orientés vers l'extérieur, similaire aux cellules de l'hémicube. L'angle solide associé à ce surfel peut être calculé dela même façon. Soit par exemple le surfel $f$ partagé par les voxels $v(a+i, b+j, c+k)$ et $v'(a+i, b+j, c+k+1)$, on peut définir

$$\widehat{A}(f) = \frac{k+0.5}{\pi(i^2+j^2+(k+0.5)^2)^{\frac{3}{2}}} \approx A(f)$$

Cette formule représente ici une approximation de l'angle solide pour la face supérieure d'un cube unité de centre $(i, j, k)$. Elle est assez semblable à un delta-facteur de forme (équation 2.12 page 53). La différence vient de ce que le facteur de forme intègre une multiplication supplémentaire, par le cosinus de l'angle avec la normale du patch « projecteur », qui rajoute un terme en $\frac{1}{\sqrt{x^2+y^2+z^2}}$.

Au final, soit $\Sigma(c)$ une surface convexe constituée de voxels, englobant $x$, et soit $F(v)$ l'ensemble des surfels extérieurs à un voxel $v \in \Sigma$, alors on peut donner de l'équation (4.6 page précédente) la forme particulière suivante :

$$B(x) = B_e(x) + \rho_d(x) \sum_{v\in\Sigma, f\in F(v), \overrightarrow{\sigma}=\overrightarrow{xf}} B(V(x, \overrightarrow{\sigma})) \cos\theta(x, V(x, \overrightarrow{\sigma}))\widehat{A}(f) \tag{4.7}$$

Cette équation peut alors être résolue avec des techniques convergentes similaires à celles déjà déployées. La section 4.2.1 page 90 en fait état.

#### 4.1.2.3 Définition de $V(x, \sigma)$

Le principe de visibilité « inverse », dans lequel la fonction $V(x, \overrightarrow{\sigma})$ permet de remplacer les facteurs de forme, est bien adapté aux voxels.

En effet, notre discrétisation de l'équation suppose qu'un et un seul voxel est visible depuis un point dans une direction donnée. Cela est difficilement admissible pour des patches, dont les tailles sont diverses. Dans une direction donnée, il est plus plausible de voir un morceau de patch plutôt qu'un patch entier, c'est ce que calculent les facteurs de forme. En revanche, la taille homogène des voxels, généralement petits, est bien adaptée à cette fonction de visibilité.

Les sections suivantes 4.2 et 4.3 montre également l'adéquation de la discrétisation par voxels avec la notion de lancer de rayons qu'implique la fonction $V(x, \overrightarrow{\sigma})$ (premier point rencontré depuis $x$ dans la direction $\overrightarrow{\sigma}$).

Notons toutefois que la fonction $V(x, \overrightarrow{\sigma})$ cache une ambiguïté dans un domaine discret : on peut trouver plusieurs droites discrètes « passant par » deux voxels $x$ et $\sigma$ (Figure 4.1). De ce fait, $V(x, \overrightarrow{\sigma})$ est mal défini sans précisions supplémentaires.

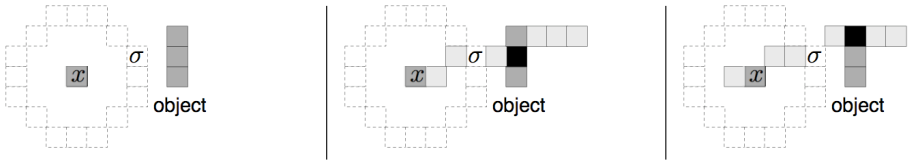



FIG. 4.1 – Plusieurs droites discrètes, même de directions similaires, peuvent passer par deux voxels $x$ et $\sigma$ donnés. $V(x, \overrightarrow{\sigma})$ est alors mal défini.

En effet, soit les pixels $x(0, 0)$ et $\sigma(3, 1)$. Le système d'inéquations d'inconnues $(a, b, \mu)$ (4.8) peut avoir plusieurs solutions. Il convient donc d'im-

poser $(a, b)$ égal à $\overrightarrow{x\sigma}$ pour fixer la direction de la droite. Cela n'est cependant pas suffisant.

$$\begin{cases} \mu \leq ax_x - by_x < \mu + \omega \\ \mu \leq ax_\sigma - by_\sigma < \mu + \omega \end{cases} \tag{4.8}$$

Si l'on fixe la direction à $(3, 1)$, un vecteur normal à la droite est $(a, b) = (1, -3)$. On fixe également l'épaisseur $\omega$ à $\max(|1|, |-3|) = 3$ pour définir une 8-connexité. Le système est alors d'inconnue $\mu$ (4.9).

$$\begin{cases} \mu \leq 1 \times 0 - 3 \times 0 < \mu + 3 \\ \mu \leq 1 \times 3 - 3 \times 1 < \mu + 3 \end{cases} \tag{4.9}$$

Ici, trois solutions sont possibles : $\mu = -2, -1$ ou $0$. En choisissant de travailler dans un repère local centré en $x$, la solution $\mu = 0$ est alors toujours possible et peut être privilégiée. En travaillant dans un repère global centré en $(0, 0, 0)$, on peut choisir le $\mu$ le plus petit.

Cette possibilité de choix ne remet pas en cause l'utilité de la fonction $\{V(x, \overrightarrow{\sigma}),\ \overrightarrow{\sigma} \in D\}$, car l'ensemble des directions $D$ représente un échantillonnage des directions de l'espace, et n'importe laquelle des droites discrètes solution pour résoudre un $V(x, \overrightarrow{\sigma})$ est convenable puisqu'elle échantillonne effectivement le champ de vision de $x$.

## 4.2 Une première approche de résolution

La discrétisation de l'équation et la première approche de résolution décrite ci-après ont été menées par Malgouyres [Mal02].

### 4.2.1 Résolution en deux étapes

Rappelons que hors optimisations, la radiosité sur les patches, faisant usage des facteurs de formes et d'une représentation matricielle, peut se résoudre en deux étapes. Soit le système linéaire dont la solution est la radiosité cherchée, où M est la matrice des facteurs de forme :

$$B = B_e + MB$$

- la première étape consiste à calculer la matrice des facteurs de forme, avec l'hémicube ou une autre méthode ;
- la seconde étape est la résolution à proprement parler, avec une suite convergente de type Jacobi.

Les optimisations apportées peuvent rendre ces deux étapes plus intimement liées, des calculs pouvant être effectués à la volée.

L'équation de radiosité que nous proposons sur des voxels fait elle aussi apparaître cette séparation. Soit l'équation (4.10) :

$$B(x) = B_e(x) + \rho_d(x) \sum_{\overrightarrow{\sigma} \in D} B(V(x, \overrightarrow{\sigma})) \cos\theta(x, V(x, \overrightarrow{\sigma}))\widehat{A}(\overrightarrow{\sigma}) \qquad (4.10)$$

- la première étape consiste à calculer pour tout $(x, \overrightarrow{\sigma})$ le voxel $V(x, \overrightarrow{\sigma})$ ;
- la seconde étape est la résolution à proprement parler, avec une suite convergente de type Jacobi.

Cette dernière peut se définir ainsi :

$$\begin{cases} B_0(x) = B_e(x) \\ B_{n+1}(x) = B_e(x) + \rho_d(x) \sum\limits_{\overrightarrow{\sigma} \in D} B_n(V(x, \overrightarrow{\sigma})) \cos\theta(x, V(x, \overrightarrow{\sigma}))\widehat{A}(\overrightarrow{\sigma}) \end{cases} \qquad (4.11)$$

### 4.2.2 Calcul de $V(x, \overrightarrow{\sigma})$

La définition de $V(x, \overrightarrow{\sigma})$ oriente naturellement vers le lancer de rayons pour en trouver la valeur. Plusieurs approches sont possibles, d'efficacités différentes.

La première approche, simpliste, consiste à résoudre le problème du ray-tracing dans le domaine continu, avec les algorithmes standards, et de trouver le point $y = V(x, \overrightarrow{\sigma})$ par des algorithmes d'intersection usuels. Ensuite, le voxel $v_y$ contenant $y$ peut être déduit. Cette méthode ne fait cependant pas usage des propriétés du repère discret. Il paraît plus judicieux de réaliser un *lancer de rayon discret*.

#### 4.2.2.1 Lancer de rayon discret

Le lancer de rayon discret consiste à représenter le rayon par une droite discrète. Il est alors possible de parcourir le rayon pas à pas depuis un voxel source jusqu'à rencontrer un voxel obstacle, qui constitue l'intersection du rayon avec un objet. La structure de données choisie pour représenter l'ensemble des voxels de la scène joue un rôle déterminant.

- la scène peut être entièrement subdivisée en voxels, lesquels indiquent s'ils contiennent ou non de la matière ;
- la scène peut être discrétisée en n'encodant des voxels que sur les surfaces ; les voxels des espaces vides sont tout simplement absents de la structure.

La deuxième représentation est évidemment moins coûteuse en espace mémoire, mais elle nécessite alors une structuration interne permettant de savoir rapidement si un voxel existe ou non. Une table de hachage peut être utilisée. L'octree est également une structure intéressante dont on peut tirer des optimisations (cf. section 4.2.2.2)

Pour la traversée du rayon, les algorithmes de parcours de droite 3D discrètes sont mis à contribution. Cela rend le lancer de rayon discret très différent du lancer de rayon usuel.

- Dans le lancer de rayon usuel, hors optimisations, on teste chaque objet et l'on garde, parmi ceux qui intersectent le rayon considéré, le plus proche de la source du rayon. Il n'y a pas de notion d'espace vide. La complexité de cet algorithme dépend du nombre d'objets et de la complexité mathématique du calcul d'intersection.
- Dans le lancer de rayon discret, on découvre le premier objet rencontré par le rayon après avoir parcouru inutilement de l'espace vide, et l'algorithme s'arrête ici. La complexité ne dépend pas du nombre d'objets de la scène mais de la distance qui sépare la source du rayon du point d'impact. Il n'y a pas de réel calcul d'intersection.

Yagel, Cohen et Kaufman ont obtenu de bons résultats avec le lancer de rayon discret [YCK92], observant comme attendu une faible complexité en fonction du nombre d'objets dans la scène, et même de la complexité mathématique de ces derniers.

#### 4.2.2.2 Optimisations du lancer de rayon

Une optimisation élémentaire du lancer de rayons usuel est l'utilisation de *bounding boxes*, des boîtes englobant grossièrement l'objet. Elles permettent de simplifier les tests d'intersection d'un rayon avec tous les objets de la scène [GA93] ; si le rayon n'intersecte pas la bounding box, pour laquelle ce calcul est très rapide, alors il ne peut intersecter l'objet. Or, les bounding boxes peuvent être stockées dans un octree pour diminuer encore le nombre de tests utiles : l'emplacement d'une bounding box dans l'octree

caractérise son emplacement dans la scène, et permet parfois d'assurer qu'il n'y aura pas d'intersection.
Cette optimisation se retrouve dans le domaine discret mais joue un rôle différent. Un octree peut être utilisé, non pour identifier des groupes d'objets, mais pour donner une représentation peu coûteuse des espaces vides. De cette façon, il devient possible d'identifier, dans la droite discrète représentant le rayon, les segments contenus dans un espace vide. Ces segments peuvent alors être ignorés dans le parcours et économiser autant de temps de traversée du rayon [SC95] (Figure 4.2).

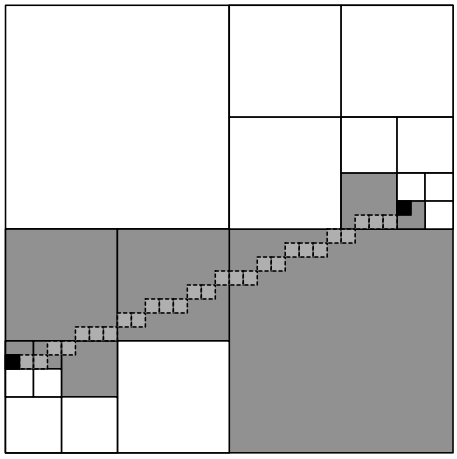


FIG. 4.2 – Dans un octree, on peut identifier des segments de droites à ignorer car situés dans un espace vide sans intérêt. Le temps nécessaire à parcourir la droite est diminué d'autant. Les voxels en noir représentent $x$ et $V(x, \overrightarrow{\sigma})$ ; la droite discrète est superposée à l'octree, dont les zones vides traversées sont colorées en gris.

### 4.2.3 Stockage des « facteurs de forme »

La résolution en deux étapes que nous avons évoquée section 4.2.1 page 90 fait une analogie avec le calcul préalable des facteurs de forme, en établissant que doivent être calculés les couples $V(x, \overrightarrow{\sigma})$. Dans la mesure où la deuxième étape, de résolution effective, nécessite au fur et à mesure des itérations plusieurs appels identiques à la fonction $V(x, \overrightarrow{\sigma})$, il est plus raisonnable de précalculer cette dernière que de réaliser à chaque fois un nouveau lancer de rayon. Ce dernier est en effet logarithmique du fait de l'accès aux voxels dans un octree.

Cela se paye en revanche par un coût de stockage important, linéaire en fonction du nombre de voxels encodant la scène et du nombre de directions échantillonnées. Cette quantité d'informations peut typiquement être déportée en mémoire secondaire, et organisée pour autoriser des chargements par

blocs en mémoire principale.

### 4.2.4 Algorithme

```
//Phase de pré-calculs
définir un ensemble de directions et les facteurs de direction associés;
pour chaque voxel x faire
    pour chaque direction σ⃗ faire
        calculer et stocker V(x, σ⃗)

//Phase de résolution
pour iterations = 1 à MaxIterations faire
    pour chaque voxel x faire
        pour chaque direction σ⃗ faire
            mise à jour de la radiosité de x en fonction de celle de
            V(x, σ⃗)
```

**Algorithme 1** : Première version de l'algorithme de radiosité

### 4.2.5 Complexité

#### 4.2.5.1 Complexité moyenne

La complexité théorique de cette première implémentation n'est pas très élevée mais son coût est assez important en pratique. Soit $N$ est le nombre de voxels de la scène, $D$ le nombre de directions considérées et $I$ le nombre d'itérations effectuées sur la suite convergente. Soit $L(N)$ la complexité moyenne de parcours de droite discrète dans le complément de la scène, pour résoudre $V(x, \overrightarrow{\sigma})$. $L(N)$ est au moins logarithmique à cause des temps d'accès à l'octree.

Le calcul de visibilité a lieu pour chaque voxel, pour chaque direction et coûte en moyenne $L(N)$. La résolution effective, pour chaque itération, accède pour chaque voxel, pour chaque direction, à $V(x, \overrightarrow{\sigma})$.
La complexité moyenne de la méthode est donc

$$\underbrace{O(N \times D \times L(N))}_{\text{pré-calcul de visibilité}} + \underbrace{O(I \times D \times N)}_{\text{résolution}}$$

#### 4.2.5.2 Résultats expérimentaux

Des résultats expérimentaux ont été conduits sur un Athlon 1.2 Ghz.

– une scène d'environ 1 000 000 de voxels dans un espace discret $315 \times 315 \times 190$, a nécessité, pour environ 9 000 directions, 25 heures de calcul. L'information de visibilité a nécessité 18Go d'espace disque.

– une scène d'environ 1 000 000 de voxels dans un espace discret $420 \times 420 \times 250$, a nécessité, pour environ 15 000 directions, 72 heures de calcul. L'information de visibilité a nécessité 50Go d'espace disque.

## 4.3 Méthode quasi linéaire de résolution

La première approche de résolution pose les bases des problématiques à résoudre, notamment l'obtention de l'information $V(x, \overrightarrow{\sigma})$ qui est la clef de la méthode. Des observations supplémentaires sur la visibilité et l'utilisation qui en est faite ont cependant permis d'améliorer largement l'algorithme. On peut obtenir une complexité quasi linéaire, et se passer de stockage en mémoire secondaire.

### 4.3.1 Observations

#### 4.3.1.1 Visibilités multiples

La première observation possible est que plusieurs visibilités peuvent être calculées sur un rayon. Si un rayon intersecte plusieurs objets, on peut en déduire différents $V(x, \overrightarrow{\sigma})$ (Figure 4.3)

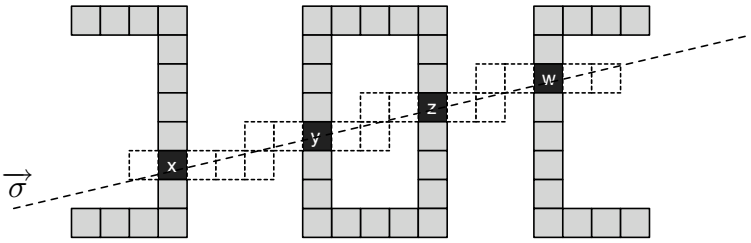



FIG. 4.3 – Le rayon de direction $\overrightarrow{\sigma}$ passant par $x$ intersecte la scène en trois autres voxels $y$, $z$ et $w$. Un seul rayon permet donc d'obtenir $y = V(x, \overrightarrow{\sigma})$, $z = V(y, \overrightarrow{\sigma})$ et $w = V(z, \overrightarrow{\sigma})$

L'information $z = V(y, \overrightarrow{\sigma})$ n'est pas pertinente car le segment $[yz]$ est à l'intérieur d'un objet, ce qui ne rentre pas en jeu dans un calcul de radiosité. Les normales en $x$, $y$, $z$ et $w$ permettent de détecter si l'on est à l'extérieur d'un objet ou non. Dans le cas d'une discrétisation trop grossière, les voxels étant plus gros que l'objet le plus fin, on peut envisager de stocker plusieurs normales par voxels pour éviter de confondre intérieur et extérieur (Figure 4.4 page suivante).

La section 4.1.2.3 page 88 abordant la définition de $V(x, \overrightarrow{\sigma})$ a mis en lumière l'ambiguïté de cette définition. On retrouve ici ses conséquences. $z = V(y, \sigma)$ révélé sur la figure 4.3 appartient à une droite discrète de direction $\overrightarrow{\sigma}$ passant par $y$. D'autres droites discrètes auraient été possibles par un léger décalage agissant sur le paramètre $\mu$ de la droite, et le $z$ trouvé aurait été différent. Le choix de ce $z$ revient en fait à fixer le paramètre $\mu$ dans un repère global et non dans un repère local à $y$.

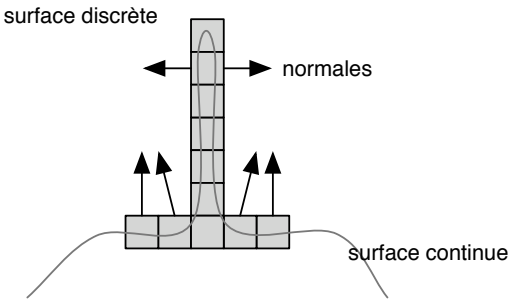



FIG. 4.4 – Si la discrétisation est trop grossière par rapport à l'objet le plus fin, on peut stocker plusieurs normales par voxel pour éviter de confondre intérieur et extérieur.

#### 4.3.1.2 Inversion du calcul de $V(x, \sigma)$

L'algorithme initial est conçu pour tester toutes les directions en chaque voxel afin d'évaluer les $V(x, \overrightarrow{\sigma})$. Or, le fait de calculer plusieurs visibilités sur un seul rayon amène à modifier le code associé. Une évolution naïve serait de garder le même code mais d'ignorer certains calculs s'ils ont déjà été faits (Algorithme 2).

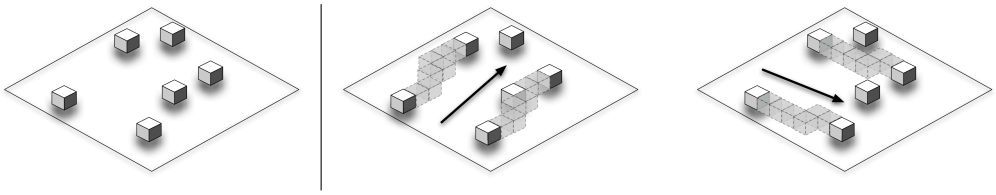
```
//Boucles de l'algorithme simple
pour chaque voxel x faire
    pour chaque direction σ⃗ faire
        calculer et stocker V(x, σ⃗)

//Évolution naïve
pour chaque voxel x faire
    pour chaque direction σ⃗ faire
        si V(x, σ⃗) n'a pas déjà été calculé alors
            calculer et stocker tous les V(y, σ⃗) à déduire des
            intersections du rayon [x, σ⃗) avec la scène.
```

**Algorithme 2** : Évolution naïve du calcul des $V(x, \sigma)$

Cet algorithme laborieux gagne à être reconsidéré. En réalité, il est plus judicieux d'inverser les boucles et de commencer par énumérer les directions. Dans une direction donnée, on peut alors identifier quels sont les rayons qui intersectent effectivement la scène, et en déduire les visibilités (Figure 4.5 page ci-contre et Algorithme 3 page suivante)

À première vue, l'algorithme a été remodelé mais pas simplifié, car la difficulté réside maintenant dans la connaissance des rayons qui intersectent la scène. Cependant, les sections 4.3.2 page ci-contre et 4.3.3 page 100 montrent comment construire efficacement ces rayons, en temps linéaire. L'algorithme



ensemble de voxels | relations de visibilité dans deux directions différentes

FIG. 4.5 – On privilégie les directions avant les voxels. La visibilité est ainsi résolue sur chaque rayon pertinent.

*//Boucles de l'algorithme simple*
**pour** *chaque voxel* $x$ **faire**
  **pour** *chaque direction* $\overrightarrow{\sigma}$ **faire**
    calculer et stocker $V(x, \overrightarrow{\sigma})$

*//Inversion du calcul*
**pour** *chaque direction* $\overrightarrow{\sigma}$ **faire**
  **pour** *chaque rayon de direction* $\overrightarrow{\sigma}$ *intersectant la scène* **faire**
    calculer et stocker tous les $V(x, \overrightarrow{\sigma})$ du rayon.

**Algorithme 3** : Inversion du calcul des $V(x, \sigma)$

de visibilité ne nécessitera alors qu'un parcours des voxels par direction, et plus aucune traversée de droite discrète 3D. La visibilité est alors obtenue en un temps optimal.

### 4.3.2 Partition de l'espace

En faisant apparaître plusieurs rayons, la figure 4.5 expose la question suivante : « étant donné une direction, puis-je savoir sur quel rayon se trouve un voxel donné ? ». La géométrie discrète apporte une réponse positive à cette question, car il est possible de partitionner l'espace en droites discrètes, de telle sorte qu'un voxel appartienne à une et une seule droite discrète.

**Proposition** :
Soit $\mathbb{Z}^3_*$ l'ensemble $\mathbb{Z}^3 \backslash \{(0,0,0)\}$. Étant donné un vecteur entier $\overrightarrow{v} \in \mathbb{Z}^3_*$, un espace discret peut être partitionné en un ensemble de droites discrètes 3D standard ou naïves, de vecteur directeur $\overrightarrow{v}$.

**Preuve** :
Nous utilisons la définition des droites discrètes utilisant deux projections [DR95] :
Soit le vecteur $(a, b, c) \in \mathbb{Z}^3$ avec $a \geq b \geq c$, une droite discrète 3D est formée

par un ensemble des points $(x, y, z) \in \mathbb{Z}^3$ tels que

$$\begin{cases} \mu \leq cx - az < \mu + \omega \\ \mu' \leq bx - ay < \mu' + \omega' \end{cases}$$

Les cas autres que $a \geq b \geq c$ sont à symétrie près.

Soit $\overrightarrow{v} = [a, b, c]$, avec $(a, b, c) \in \mathbb{Z}^3_*$, $(a, b, c)$ étant premiers entre eux. On suppose sans perte de généralité que $a \geq b \geq c \geq 0$.
Une droite 3D discrète de vecteur directeur $\overrightarrow{v}$ est définie par deux projections. La connexité de cette droite 3D est liée à celle de ses projections. Nous pouvons étudier séparément les cas naïf (8-connexité) et standard (4-connexité). L'épaisseur arithmétique $\omega$ étant notée :

$$\text{cas naïf} : \begin{cases} \omega_{ab} = \max(|a|, |b|) \neq 0 \\ \omega_{ac} = \max(|a|, |c|) \neq 0 \\ \omega_{bc} = \max(|b|, |c|) \end{cases} \qquad \text{cas standard} : \begin{cases} \omega_{ab} = |a| + |b| \neq 0 \\ \omega_{ac} = |a| + |c| \neq 0 \\ \omega_{bc} = |b| + |c| \end{cases}$$

Puisque l'on a $a \geq b \geq c \geq 0$, les projections 2D à considérer se situent dans les plans (Oxy) et (Oxz). C'est pourquoi seul $\omega_{bc}$ peut être nul. Ainsi, la droite 3D discrète est équivalente à l'ensemble des $(x, y, z)$ tels que :

$$\begin{cases} \mu_{ab} \leq -bx + ay < \mu_{ab} + \omega_{ab} \\ \mu_{ac} \leq -cx + az < \mu_{ac} + \omega_{ac} \end{cases}$$

Dans la mesure où $a, b, c, \omega_{ab}, \omega_{ac}$ sont donnés, seule la position de la droite peut-être choisie. Nous nommons donc $L(\mu_{ab}, \mu_{ac})$ l'ensemble des voxels de la droite. On introduit alors l'ensemble des droites discrètes suivant :

$$\{L_{i,j}\}_{(i,j)\in\mathbb{Z}^2} \;=\; \{L(i \times \omega_{ab}, j \times \omega_{ac})\}_{(i,j)\in\mathbb{Z}^2}$$

Soit $i$ et $j$, une droite 3D discrète $L_{i,j}$ est définie par :

$$\begin{cases} i \times \omega_{ab} \leq -bx + ay < i \times \omega_{ab} + \omega_{ab} \\ j \times \omega_{ac} \leq -cx + az < j \times \omega_{ac} + \omega_{ac} \end{cases} \tag{4.12}$$

ou encore

$$\begin{cases} i \times \omega_{ab} \leq -bx + ay < (i+1) \times \omega_{ab} \\ j \times \omega_{ac} \leq -cx + az < (j+1) \times \omega_{ac} \end{cases} \tag{4.13}$$

**Appartenance :** Soit un voxel $(x, y, z) \in \mathbb{Z}^3$, ce voxel appartient à $L_{k,l}$ avec $k = \lfloor \frac{-bx+ay}{\omega_{ab}} \rfloor$ et $l = \lfloor \frac{-cx+az}{\omega_{ac}} \rfloor$ (où $\lfloor x \rfloor$ est la partie entière de $x$).

**Unicité :** Les $L_{i,j}$ sont disjoints deux à deux et constituent une partition. Par l'absurde : soit un voxel $v = (x, y, z)$ appartenant à $L_{i,j}$ et $L_{i',j'}$, avec $(i, j) \neq (i', j')$. On suppose par exemple $i < i'$, car si $i = i'$, le raisonnement suivant tient toujours en remplaçant $i$ par $j$ en choisissant $j < j'$.

$$\begin{aligned} v \in L_{i,j} \text{ et } v \in L_{i',j'} &\Rightarrow \begin{cases} i \times \omega_{ab} \leq -bx + ay < (i+1) \times \omega_{ab} \\ i' \times \omega_{ab} \leq -bx + ay < (i'+1) \times \omega_{ab} \end{cases} \\ &\Rightarrow \begin{cases} -bx + ay < (i+1) \times \omega_{ab} \leq i' \times \omega ab \\ i' \times \omega_{ab} \leq -bx + ay \end{cases} \\ &\Rightarrow -bx + ay < -bx + ay \qquad \text{ce qui est impossible} \end{aligned} \tag{4.14}$$

En conclusion, étant donné une direction, et une connexité standard ou naïve, l'ensemble des $(L_{i,j})_{i,j}$ constitue une partition de l'espace discret en droites 3D discrètes de même direction. De plus, une opération élémentaire utilisant la fonction « partie entière » permet de déduire, par les coordonnées d'un voxel $x$, le couple $(i, j)$ tel que $x \in L_{i,j}$.

### 4.3.3 Construction des rayons

La propriété d'appartenance d'un voxel à un unique rayon, établie dans la section 4.3.2 page 97, permet de résoudre efficacement le problème de visibilité. Dans une direction $\overrightarrow{\sigma}$ donnée, chaque rayon peut être représenté par une liste des voxels qu'il contient, et ces listes peuvent être stockées dans un tableau bi-dimensionnel indexé sur les indices $i$ et $j$ caractérisant le rayon. Deux voxels successifs dans une liste sont alors un couple $(x, y)$ tel que $y = V(x, \overrightarrow{\sigma})$ (Figure 4.6).

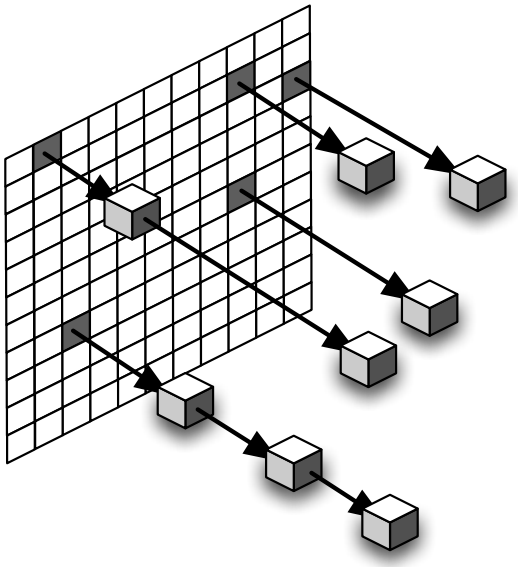


FIG. 4.6 – Pour une direction donnée, les rayons peuvent être vus comme les listes des voxels qu'ils intersectent. Ces listes sont référencées dans un tableau 2D indexé sur les indices $i$ et $j$ caractéristiques du partitionnement.

On évite ainsi d'avoir à effectuer une traversée des droites 3D discrètes : il suffit de construire tous les rayons en même temps en parcourant une fois tous les voxels. Cela revient à inverser deux boucles dans l'algorithme initial (Algorithme 4 page suivante).

La difficulté réside maintenant dans la mise à jour des listes. Les voxels y sont *a priori* insérés dans un ordre quelconque. Or, chaque liste doit être triée pour respecter les emplacement des voxels les uns par rapport aux autres dans l'espace (Figure 4.7 page ci-contre).

*//Boucles de l'algorithme initial*
**pour** *chaque voxel* $x$ **faire**
  **pour** *chaque direction* $\overrightarrow{\sigma}$ **faire**
    calculer et stocker $V(x, \overrightarrow{\sigma})$

*//Inversion des boucles*
**pour** *chaque direction* $\overrightarrow{\sigma}$ **faire**
  **pour** *chaque voxel* $x$ **faire**
    placer $x$ dans sa liste $L(i, j)$
  //Les listes encodent la visibilité dans la direction $\overrightarrow{\sigma}$

**Algorithme 4** : Inversion des boucles pour la construction des rayons

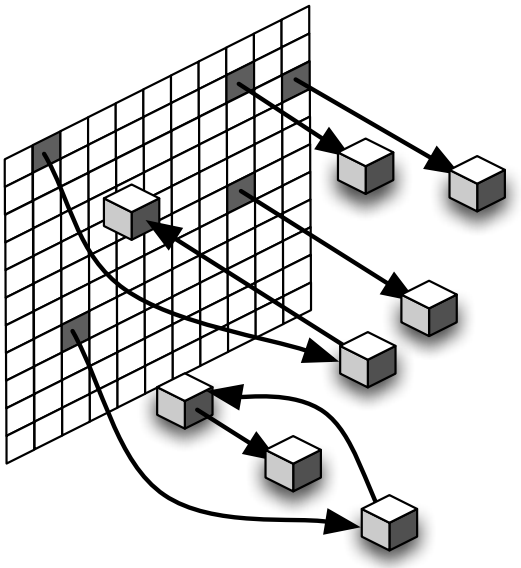


FIG. 4.7 – Les listes doivent être triées, car leur ordre de remplissage quelconque ne respecte pas forcément la visibilité des voxels.

#### 4.3.3.1 Approche non optimale par ajout de l'étape de tri

Le critère de tri n'est pas difficile à exprimer : pour deux voxels $x$ et $y$, dans la direction courante $\overrightarrow{\sigma}$, le produit scalaire $\overrightarrow{xy}.\overrightarrow{\sigma}$ suffit à positionner $x$ par rapport à $y$. Cependant, comme toutes les listes doivent être triées, cela est associé à un coût incompressible de $O(N \log N)$, $N$ étant le nombre de voxels répartis dans toutes les listes.

Ce tri peut être réalisé en post-traitement (Figure 4.8 et Algorithme 5), ou à la volée lors de l'insertion dans la liste. La complexité globale reste la même : $O(N \log N)$.

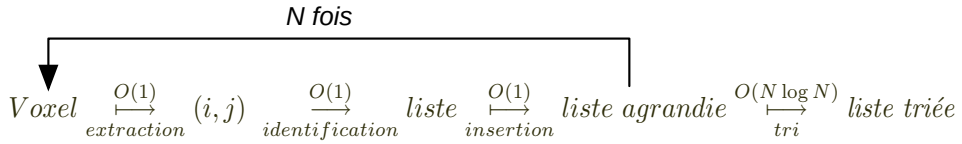



FIG. 4.8 – La construction des rayons nécessite un tri en post-traitement, pour refléter les positions respectives des voxels sur chaque rayon.

```
construire l'ensemble des directions;
pour chaque direction faire
    calculer et stocker le facteur de direction;
//le nombre d'itérations est une faible constante
pour iterations = 1 à MaxIterations faire
    //le nombre de directions est une grosse constante
    pour chaque direction faire
        //étape de répartition : O(N)
        pour chaque voxel faire
            l'ajouter en fin de la liste qui représente le rayon (droite
            3D discrète) auquel il appartient;
        //étape de tri : O(N log N)
        pour chaque liste faire
            trier les voxels de chaque liste pour refléter leurs positions
            respectives;
        //étape de propagation (résolution) : O(N)
        pour chaque liste faire
            propager la radiosité entre les voxels contigus, en mettant
            à jour B_{n+1};
        remise à zéro des listes;
```

**Algorithme 5** : Algorithme avec tri explicite des listes

#### 4.3.3.2 Approche optimale grâce à un tri lexicographique

Les listes ont besoin d'être triées car les voxels qui les remplissent sont parcourus dans un ordre quelconque. Mais si cet ordre n'est pas quelconque, on peut obtenir un tri naturel des listes. La figure 4.9 en donne un exemple. Les voxels, numérotés de 1 à 7, sont parcourus dans un ordre particulier, de telle sorte que les listes construites lors du parcours soient naturellement triées.

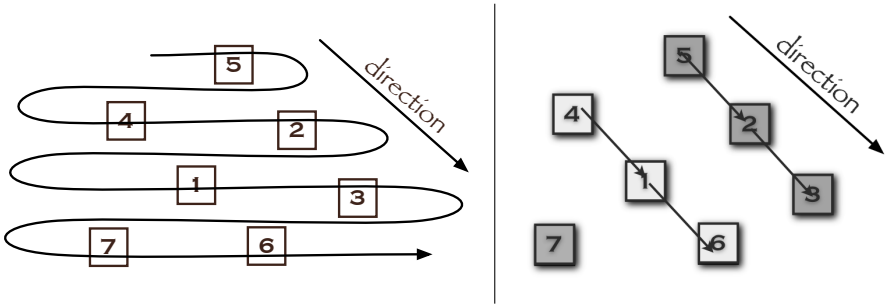



FIG. 4.9 – Les voxels, numérotés de 1 à 7, sont parcourus dans un ordre particulier, de telle sorte que les listes construites lors du parcours soient naturellement triées.

Un sorte de front d'onde, perpendiculaire à la direction courante, et se déplaçant dans cette direction, rencontre les voxels dans leur ordre naturel pour la direction courante. Dans le cas discret, ce front d'onde peut être aligné selon un des axes de coordonnées. Enfin, ce front d'onde peut être implémenté sous la forme d'un ordre lexicographique (Figure 4.10).

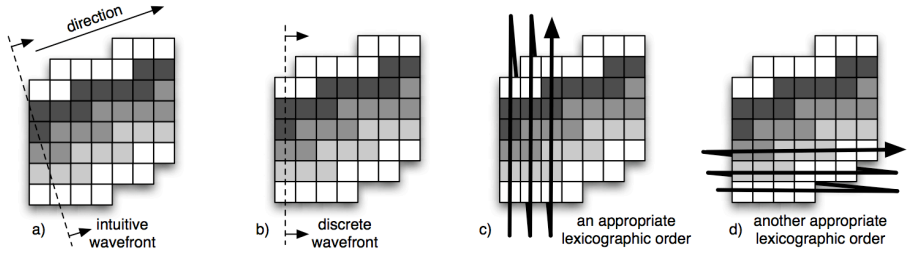



FIG. 4.10 – Un parcours des voxels dans un ordre approprié conserve leur ordre naturel dans les droites discrètes. a) Un front d'onde rencontre les voxels dans le bon ordre. b) Dans le cas discret, ce front d'onde peut être aligné sur un des axes de coordonnées. c) et d) Ce front d'onde peut être représenté par un ordre lexicographique sur les coordonnées.

L'apparition d'un ordre lexicographique n'est pas étonnant. Hormis le cas de la bulle, traité à part en section 4.3.3.5 page 106, une droite 3D discrète

peut être intuitivement traversée dans un ordre lexicographique bien choisi en fonction du vecteur directeur. Ce choix est dirigé par l'algorithme suivant :

- soit $\overrightarrow{\sigma} = (a, b, c)$ la direction courante.
- si $a$ (resp. $b$, $c$) est positif (resp. négatif), $x$ (resp. $y$, $z$) est pris en ordre croissant (resp. décroissant)
- l'ordre respectif de $x$, $y$, $z$ dépend de l'ordre des valeurs absolues de $a$, $b$, $c$ ; on privilégie les plus élevées.

Exemple : pour la direction $(a, b, c) = (3, -5, 2)$, on choisit l'ordre lexicographique $y \downarrow x \uparrow z \uparrow$, car $|b| \geq |a| \geq |c|$ ($\downarrow$ représente un ordre décroissant, $\uparrow$ un ordre croissant).

Jusque-là, aucune condition n'était requise sur la structure de données fournissant les voxels à l'algorithme de radiosité, ils étaient parcourus dans un ordre quelconque. On exige maintenant de cette structure qu'elle puisse nous fournir les voxels dans un ordre lexicographique donné. Il y a en 3D 48 ordres lexicographiques possibles, selon l'ordre respectif de $x$, $y$, $z$, et qu'on les considère croissants ou décroissants. La section 4.3.3.4 page 106 montre comment réduire ce nombre à 4.

La contrainte de pouvoir obtenir les $N$ voxels en ordre lexicographique impose un tri, en $O(N \log N)$. Ce tri n'est cependant plus effectué en post-traitement de la construction des listes, mais en pré-calcul à l'algorithme total, il est valide pour tous $\overrightarrow{\sigma}$. Le coût de $O(N \log N)$ n'est donc pas multiplié par les constantes des boucles de l'algorithme (Algorithme 6 page suivante). La complexité spatiale est quant à elle linéaire : chaque ordre lexicographique peut être implémenté comme un tableau trié de pointeurs sur les voxels.

#### 4.3.3.3 Complexités

**Complexité temporelle** :
L'algorithme ainsi décrit dépend de plusieurs paramètres. Le nombre $N$ de voxels, le nombre $I$ d'itérations (moins d'une dizaine) et le nombre $D = |\Sigma|$ de directions (plusieurs centaines). La complexité temporelle obtenue est donc :

$$\underbrace{4 \times O(N \log N)}_{\substack{\text{préparation des ordres} \\ \text{lexicographiques}}} + \underbrace{I \times D \times (O(N) + O(N))}_{\text{résolution de la radiosité}} = \underbrace{O(N \log N)}_{\substack{\text{négligeable} \\ \text{en pratique}}} + O(I \times D \times N)$$

Les constantes $I$ et $D$ sont apparentes pour montrer qu'elles ne concernent que la partie linéaire de l'algorithme. En pratique, le tri en $O(N \log N)$ est l'affaire d'une fraction de secondes, tandis que la résolution est de l'ordre de minutes ou d'heures.

```
préparer les ordres lexicographiques requis : O(N log N);

Construire l'ensemble des directions;
pour chaque direction faire
    calculer et stocker le facteur de direction;

//le nombre d'itérations est une faible constante
pour iterations = 1 à MaxIterations faire
    //le nombre de directions est une grosse constante
    pour chaque direction faire
        //étape de répartition : O(N)
        sélectionner l'ordre lexicographique approprié;
        pour chaque voxel (dans l'ordre lexicographique) faire
            //Les voxels sont triés naturellement à l'insertion
            l'ajouter en fin de la liste qui représente le rayon (droite
            3D discrète) auquel il appartient;
        //étape de propagation (résolution) : O(N)
        pour chaque liste faire
            propager la radiosité entre les voxels contigus, en mettant
            à jour B_{n+1};
        remise à zéro des listes;
```

**Algorithme 6** : Algorithme quasi linéaire

**Complexité spatiale** :
On suppose que la scène est constituée de $N$ voxels encodés dans une structure en $O(N)$. Dans une direction, l'ensemble de toutes les listes contient une référence unique à chaque voxel, si bien que la complexité spatiale de cet ensemble est de $O(N)$. Les ordres lexicographiques précalculés sont stockés dans des tableaux et coûtent $O(N)$.

Un tableau bi-dimensionnel est utilisé pour stocker les listes. Ces dernières formant une partition de l'espace, le nombre de listes est lié au carré de la largeur de la scène. Seules les surfaces des objets étant discrétisées, cette largeur est en moyenne en $O(\sqrt{N})$. $O(N)$ listes sont donc nécessaires, éventuellement vides. Les listes sont remises à zéro à chaque direction, le coût de stockage n'augmente donc pas au fur et à mesure du déroulement de l'algorithme.

Au final, la complexité spatiale est un $O(N)$ qui peut être confondu avec le $O(N)$ nécessaire pour stocker les données à traiter.

#### 4.3.3.4 Réduire le nombre d'ordres lexicographiques

Parmi les 48 ordres lexicographiques possibles, on peut n'en calculer que 24, les autres étant simplement les ordres opposés. En stockant les ordres dans des tableaux, il suffit de parcourir ces tableaux de la fin vers le début pour obtenir les ordres opposés.

De plus, on peut constater que l'ordre respectif de $x$, $y$, et $z$ n'a en réalité pas d'incidence dans notre construction. La figure 4.10 page 103 en est un exemple en 2D : considérer $x$ avant $y$ ou $y$ avant $x$ ne change rien aux listes construites. En 3D, on peut énumérer les différentes configurations de changement de palier sur la droite ($\pm 1$ en $x$, $y$ ou $z$), et observer que les ordres respectifs de $x$, $y$ et $z$ ne modifient pas la traversée de la droite naïve ou standard. On peut donc se contenter de 4 ordres lexicographiques pour prendre en compte tous les cas possibles de direction $(a, b, c)$. On peut retenir par exemple $(x \uparrow y \uparrow z \uparrow)$, $(x \downarrow y \uparrow z \uparrow)$, $(x \uparrow y \downarrow z \uparrow)$ et $(x \uparrow y \uparrow z \downarrow)$, avec les notations de la section 4.3.3.2 page 103.

Les bulles sont cependant un cas problématique, car il subsiste une ambiguïté sur certains voxels, détaillée dans la section suivante.

#### 4.3.3.5 Cas de la bulle

Dans le cas particulier d'une bulle sur une droite 3D discrète, certains voxels n'ont pas d'ordre naturel (Figure 4.11). L'ordre lexicographique choisi agit donc de façon arbitraire dans ce cas.

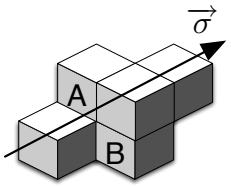



FIG. 4.11 – Une bulle sur une droite 3D discrète. Dans la direction $\overrightarrow{\sigma}$, A et B n'ont pas d'ordre naturel.

Le cas des bulles est en pratique assez peu problématique. On peut effectuer les observations suivantes :

- le cas est plutôt rare. Tous les voxels de la droite discrète n'existent pas forcément dans la scène, il faut la coïncidence d'existence et d'emplacement pour créer cette situation.
- $A$ et $B$ étant très proches, leur normales sont vraisemblablement peu différentes. Pour un voxel en amont ou en aval, l'échange d'énergie est quasi équivalent.
- une erreur générée dans cette direction est sans doute largement compensée dans d'autres directions.

– si, dans le cas d'une forte courbure présente à l'emplacement de la bulle, les normales en $A$ et $B$ sont très différentes, un échange malencontreux de $A$ et $B$ risque d'empêcher un échange correct d'énergie avec un éventuel voxel en amont ou en aval. Le bord d'un objet est un exemple (Figure 4.12). Dans ce cas, on peut éventuellement détecter un tel échange malencontreux en observant les normales en $A$ et $B$ par rapport à des voxels en amont ou en aval. Le plus souvent, deux normales successives doivent avoir un produit scalaire négatif. Même si cela n'est pas systématique, c'est un bon moyen de corriger les éventuelles erreurs de bulle.

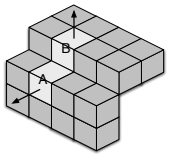



FIG. 4.12 – Le bord d'un objet est un emplacement contrariant pour une bulle, car les normales sont très différentes

Nous avons également effectué un comptage des cas dégénérés dus aux bulles sur une scène discrétisée de plus en plus finement, *cabin.mgf*[1]. Par rapport à la taille de la scène, une discrétisation en environ 100 000 voxels commence d'être raisonnable. Les cas dégénérés sont ceux où deux voxels $A$ et $B$ ne sont pas ordonnés par l'ordre lexicographique (Figure 4.11 page ci-contre). Nous avons utilisé un grand nombre de directions (16 418) pour maximiser les cas possibles. Les chiffres relevés correspondent au pourcentage de bulles par rapport au nombre de voxels, la moyenne étant faite pour toutes les directions. Nous avons également fait ces calculs uniquement sur les cas où les deux voxels ont une normale opposée (Figure 4.13 page suivante). Ces cas nécessitant éventuellement une intervention pour réordonner les voxels sont en proportion constante mais faible (moins de 0,2% des voxels pour une discrétisation correcte).

### 4.3.4 Calcul de l'ensemble des directions

La section 4.1.2.2 page 87 a introduit une définition de l'ensemble $D$ des directions $\overrightarrow{\sigma}$ choisies, comme l'ensemble des surfels extérieurs d'une surface $\Sigma$. Un tel ensemble doit être assez isotrope, et paramétré de façon à générer plus ou moins de directions. Par analogie avec les cellules de l'hémicube, on pourrait choisir un cube comme surface $\Sigma$. Mais les surfels étant des morceaux de plans, une sphère discrète convient tout aussi bien. Le rayon de

[1] scène classique disponible à l'URL `http://radsite.lbl.gov/mgf/scene/cabin.mgf.Z`

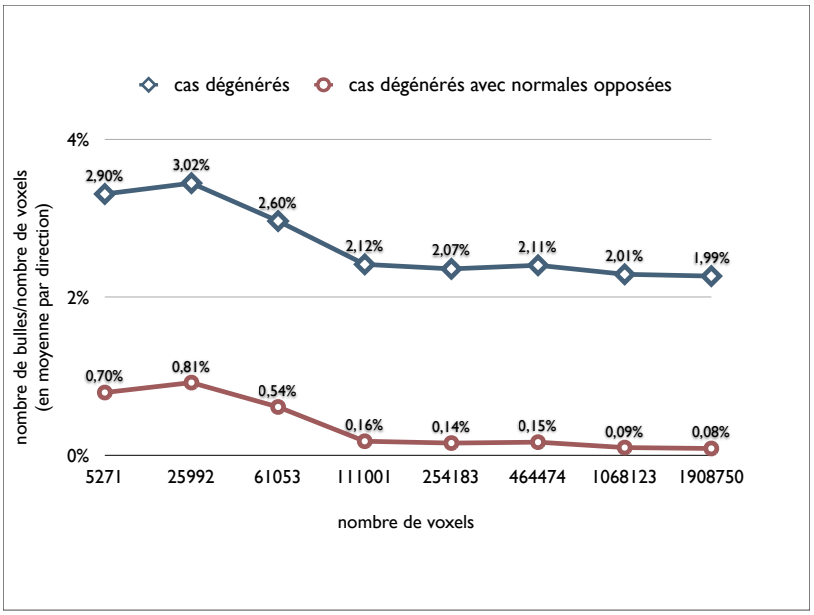
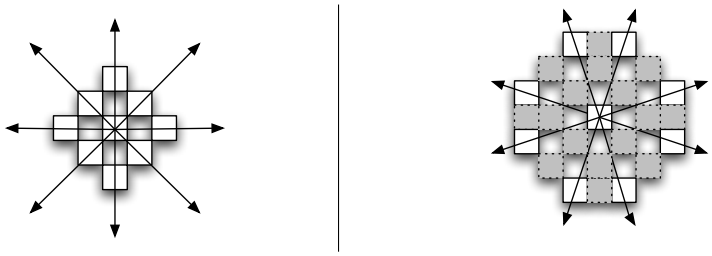



FIG. 4.13 – Les cas nécessitant éventuellement une intervention pour réordonner les voxels (en cas de normales opposées) sont en proportion constante mais faible (moins de 0,2% des voxels pour une discrétisation correcte).

la sphère sert alors à paramètrer le nombre de directions. Celui-ci augmente avec le carré du rayon (Figure 4.14 page suivante).

Pour assurer une bonne convergence de l'algorithme de radiosité, il peut être intéressant de considérer les directions dans un ordre aléatoire. La lumière est ainsi répartie plus rapidement dans divers coins de la scène. Une autre approche non aléatoire consiste à construire un ensemble de directions de façon itérative, en augmentant la taille de la sphère. Les premières directions effectuent une répartition grossière de la radiosité dans quelques directions, tandis que les directions suivantes affinent la répartition. Dans ce cas, il faut faire attention à ne pas générer plusieurs fois la même direction, et à évaluer les facteurs de direction de l'ensemble final, et non pas de chaque sous-ensemble intermédiaire.



sphère englobante de petit rayon | sphère englobante de plus grand rayon

FIG. 4.14 – L'ensemble des directions peut être construit incrémentalement en augmentant le rayon de la sphère discrète. Dans ce cas, les directions sont utilisées dans un ordre qui raffine le calcul. Il faut prendre garde à ne pas répéter des directions déjà calculées.

### 4.3.5 Optimisations

#### 4.3.5.1 Optimisations techniques

L'implémentation des rayons sous forme de listes laisse place à quelques optimisations purement techniques qui ont toutefois une bonne efficacité.

Tout d'abord, plutôt que d'utiliser des listes chaînées de voxels, les rayons peuvent être encodés dans des tableaux de capacité pré-allouée. L'ajout d'un élément en queue reste une opération en temps constant, mais le parcours de cette « liste » est un peu plus rapide.

Ensuite, ces listes doivent être indexées selon deux paramètres $i$ et $j$. Pour utiliser un tableau bi-dimensionnel, il faut connaître les valeurs minimum et maximum de $i$ et $j$, qui peuvent être calculées en testant toutes les directions aux huits coins d'un parallélépipède englobant la scène. On peut aussi éviter d'utiliser un tableau bi-dimensionnel et lui préférer une table de hachage de clef $(i, j)$.

Pour éviter la création de bulles, on pourrait imaginer utiliser la définition des droites discrètes à trois projections. Dans ce cas, les listes ne pourraient plus être stockées dans un tableau bi-dimensionnel, mais il faudrait bien utiliser une table de hachage de clef $(i, j, k)$.

Lors de la phase de propagation dans la direction $\overrightarrow{\sigma}$, on parcourt les listes du début vers la fin. En effectuant le parcours inverse, on peut effectuer la propagation dans la direction -$\overrightarrow{\sigma}$. On peut ainsi faire l'économie de la moitié des phases de répartition en parcourant les listes dans les deux sens.

#### 4.3.5.2 Stockage de la première itération

Une des forces du nouvel algorithme de radiosité est sa capacité à générer la visibilité à la volée, sans nécessiter de pré-calcul coûteux. En revanche, on constate que les mêmes calculs sont effectués plusieurs fois, car chaque itération utilise les mêmes informations de visibilité que la précédente. On peut donc envisager de stocker en mémoire secondaire les visibilités générées lors de la première itération. Cela est coûteux en espace car il faut stocker les résultats pour chaque direction. Les itérations suivantes peuvent alors économiser l'étape de répartition des voxels dans les listes en restaurant l'état désiré depuis la mémoire secondaire.

En pratique, nous avons observé un ralentissement de 20% lors de la première itération, en raison du stockage, tandis que les itérations suivantes se voyaient accélérées de 30%. D'un côté, des ressources supplémentaires ont permis d'améliorer le temps de calcul. Mais le coût en espace est très élevé comparé au gain de performance relevé. Il était de 60 Go pour $2 \times 10^6$ voxels et 15 000 directions.

## 4.4 Résultats expérimentaux

La figure 4.15, reprise de [Mal02], est constituée de 310 000 patches, et a été discrétisée en approximativement $2 \times 10^6$ voxels. La radiosité a été calculée dans les mêmes conditions que l'algorithme initial : 6 itérations, environ 15 000 directions, sur un Athlon 900 MHz avec 1,5 Go de RAM. Le temps de calcul a été amélioré de 60%, descendant de 72 à 27 heures de calcul pour un résultat visuel identique. Aucun espace disque n'a été nécessaire pour ce calcul entièrement réalisé en mémoire centrale.

FIG. 4.15 – L'éclairage de cette scène est le résultat de l'algorithme de radiosité sur les voxels de sa représentation discrète.

Ce résultat expérimental de 27 heures est un temps de calcul bien plus élevé que ce que donnerait un algorithme classique de radiosité par patches, de l'ordre de quelques minutes pour une telle scène. Cela est dû au fait que les patches, qui peuvent être de tailles variées et dynamiques, sont souvent mieux adaptés à la représentation des grandes surfaces que les voxels, qui sont de tailles identiques. Ici, pour capturer les détails de cette scène, les voxels doivent être petits, ce qui augmente considérablement leur nombre (310 000 patches, 2 millions de voxels !). De plus, la scène étant peu dense, les listes de voxels sont très creuses. Elle a cependant été choisie pour témoigner de l'amélioration de l'algorithme de radiosité par rapport à sa version antérieure. La section 4.5 page 114 donne des explications quant à ces limitations apparentes.

Sur un serveur de calcul Bi-Xeon $2 \times 2,4$ GHz, nous avons également observé les temps de calcul pour la scène *cabin.mgf*[1], discrétisée de plus en plus finement (Figure 4.16).

Les courbes n'ont pas été tracées pour des nombres différents d'itérations, car le temps de calcul est parfaitement stable d'une itération sur l'autre. Cela était attendu, dans la mesure où les calculs de chaque itération ne diffèrent que des quantités de radiosité transmises de voxel à voxel. Le temps de calcul total observé est bien le temps de la première itération multiplié par le nombre d'itérations.

On peut observer la linéarité du temps de calcul en fonction du nombre de voxels (Figure 4.17 page suivante) ou du nombre de directions (Figure 4.18).

Le rapport *temps/nombre de voxels* n'est pas tout à fait constant. Cela doit être dû aux coûts d'allocations/désallocations mémoire effectués lors de la construction/destruction des listes de voxels.

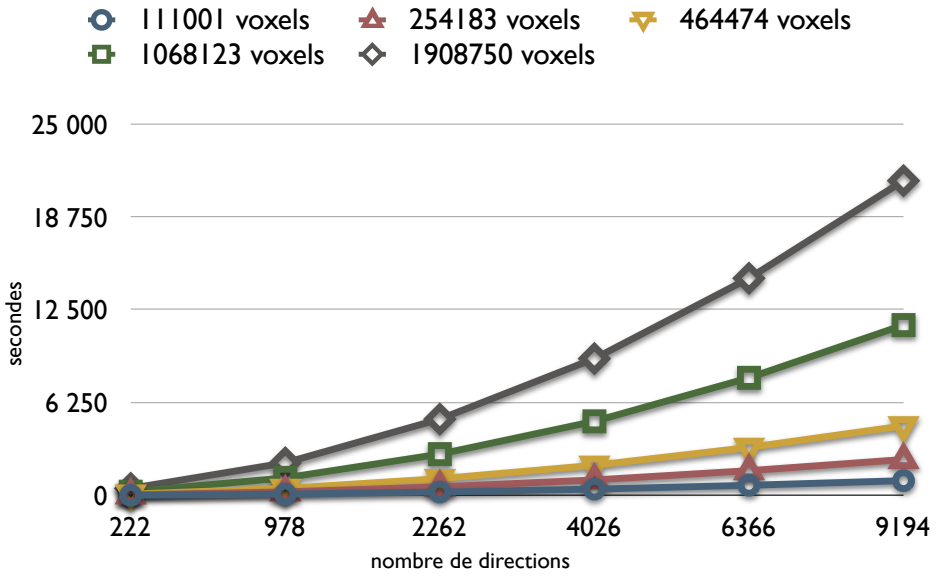



FIG. 4.16 – Temps de calcul observés pour une scène discrétisée de plus en plus finement.

[1] scène classique disponible à l'URL `http://radsite.lbl.gov/mgf/scene/cabin.mgf.Z`

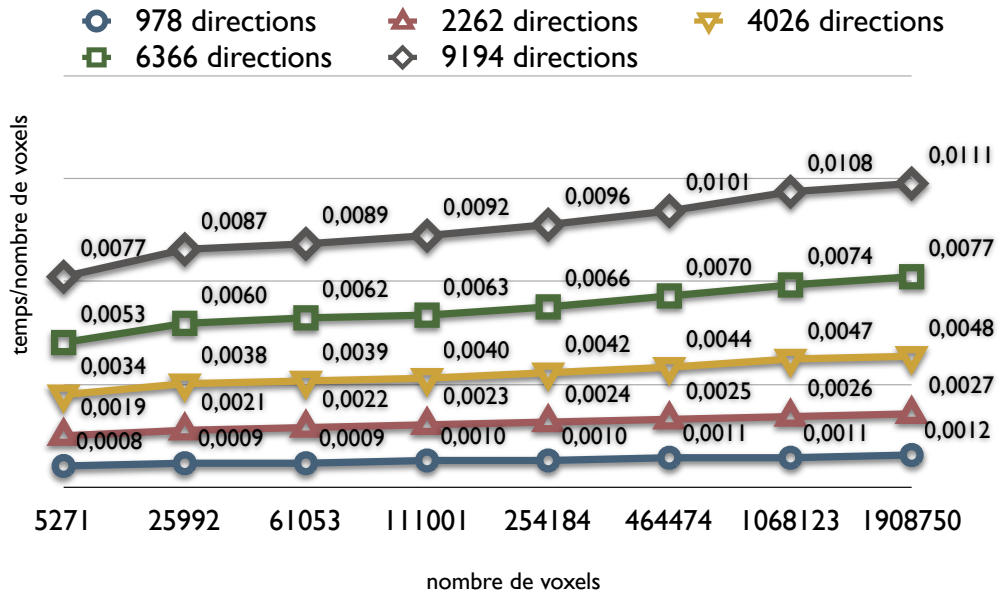



FIG. 4.17 – On peut observer la quasi-constance du rapport temps de calcul/nombre de voxels. L'augmentation de la constante doit être dû aux coûts d'allocations/désallocations mémoire effectués lors de la construction/destruction des listes de voxels.

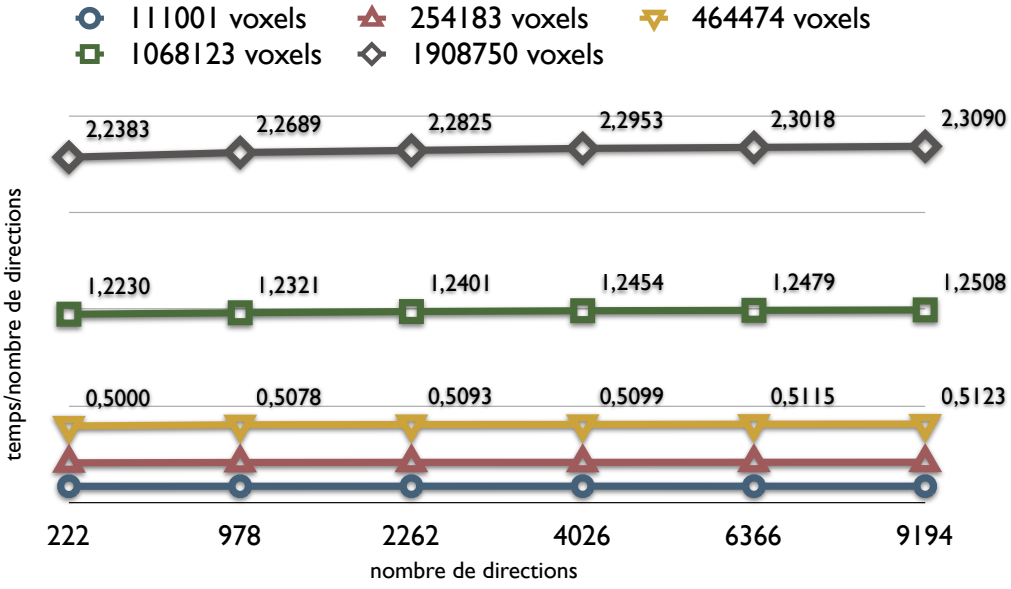



FIG. 4.18 – On peut observer la constance du rapport temps de calcul/nombre de directions

## 4.5 Limitations

L'algorithme présenté montre sa capacité à utiliser les voxels pour résoudre le problème de la visibilité, et ainsi donner une solution à l'équation de radiosité. Cependant, on peut formuler quelques critiques en le comparant aux possibilités offertes par les algorithmes par patches.

### 4.5.1 Nombre et taille des voxels

Tout d'abord, malgré la complexité théorique faible, le calcul reste lent pour un nombre de voxels élevé. Or, une surface nécessite généralement beaucoup de voxels pour être représentée, et le nombre d'éléments modélisant la scène est plus important que dans le cas de patches. Une problématique proche du nombre de voxels est le choix de leur taille. Contrairement aux patches, les voxels sont de taille fixe et unique. Capturer des détails finement dans une partie de la scène implique de choisir une petite taille, ce qui provoque une multiplication du nombre de voxels dans toute la scène, même dans les zones de moindre précision requise. Or, l'échantillonnage des directions doit être cohérent avec la taille des voxels, de petits voxels nécessitant plus de directions pour une bonne propagation de l'énergie. Cela alourdit d'autant le calcul.

Si les voxels pouvaient être de tailles diverses, l'algorithme de visibilité devrait être profondément remanié. Il s'agirait de fixer une épaisseur de droite discrète et de détecter pour chaque voxel la ou les droites qui le traversent. Un voxel pourrait donc être attribué à plusieurs rayons, ce qui nécessiterait d'introduire des calculs par intervalles. Cela semble réalisable en conservant des calculs arithmétiques simples de géométrie discrète, mais nous n'avons pas effectué de recherches dans ce sens.

La taille variable des voxels soulève également la question de la subdivision hiérarchique.

### 4.5.2 Subdivision hiérarchique

En autorisant les patches à être de tailles variées, il est possible de les fusionner ou les subdiviser afin d'adapter la finesse du modèle à la précision requise. En conservant une information hiérarchique de parenté entre les patches, la propagation de l'énergie peut ainsi se faire de façon plus ou moins « globale » sur une surface donnée (cf. section 2.4.2 page 58).

Cela paraît difficile à établir pour les voxels, dont la taille est choisie fixe (cf. section précédente). Un voxel étant cubique, une augmentation de sa taille risque d'épaissir artificiellement une surface et de provoquer des erreurs importantes dans le calcul de visibilité.

### 4.5.3 Choix d'un domaine adapté

Il ne faut cependant pas perdre de vue le champ d'application de l'algorithme, et les limitations évoquées ci-dessus doivent être replacées dans leur contexte.

Tout d'abord, il est normal que le calcul reste lourd avec un grand nombre de voxels et de directions, mais l'intérêt est surtout de le voir rester abordable dans ce cas. La complexité étant faible, on peut se permettre une discrétisation fine pour espérer un résultat numériquement meilleur. Les temps de calcul sont stables et prévisibles, on dispose d'un très bon contrôle de la paramétrisation. De plus, le chapitre 6 exposera les facilités de parallélisation de l'algorithme, qui lui donnent un nouvel intérêt pour les traitements volontairement lourds.

Ensuite, les voxels étant des entités volumiques, ils ne sont pas toujours adaptés à la représentation des surfaces, ou plutôt, ne peuvent disposer des mêmes propriétés que les patches. Un grand mur plat où la luminosité varie peu est peu efficacement représenté par des voxels, mais traduit certainement une scène très « creuse ». C'est pour les scènes plutôt denses, nécessitant des petits patches, que les voxels sont compétitifs.

Un modèle de nuage météorologique (cf. section 6.3.2 page 149) est un exemple idéal de scène volumique par essence. Il est difficile de représenter un nuage avec des surfaces, et c'est bien là que des briques 3D peuvent être utilisées, fusionnées ou éclatées avec succès.

L'algorithme de radiosité présenté dans la forme actuelle ne doit donc pas être limité à l'image de synthèse traditionnelle faisant grand usage des surfaces. Il est aussi une démonstration de faisabilité ouvrant des possibilités sur les domaines volumiques.

## 4.6 Conclusion

Nous avons pu, en le remodelant et en utilisant des propriétés des droites discrètes, améliorer l'algorithme de radiosité sur les voxels de façon importante. Tout en démontrant sa viabilité, nous avons obtenu une complexité temporelle quasi linéaire et une complexité spatiale linéaire. Cela semble optimal vu le problème posé.

Malgré les limitations inhérentes au choix des voxels plutôt que des patches, l'algorithme bénéficie d'avantages, tels la capacité à gérer une lourde charge, et des temps de calcul stables et prévisibles.

Exploiter au mieux l'algorithme passe par le choix de scènes plutôt denses. La forme de l'algorithme lui ouvre également la porte à quelques extensions. Des capacités de parallélisation peuvent notamment être déployées. Mais c'est dans le chapitre 6 que l'exploitation de ces extensions est présentée.

Le but recherché a été atteint : utiliser au mieux la géométrie discrète pour propager de l'information entre les voxels d'un espace discret. L'utilisation de droites discrètes pour résoudre le problème de la visibilité a montré un intérêt dans le cas de la radiosité, où de nombreuses directions doivent être considérées.

# Chapitre 5

# Bidirectional Reflectance Distribution Functions

La notion de BRDF a été introduite en section 1.2.3 page 27 parmi les différents modèles d'éclairement possibles. L'obstacle majeur à l'utilisation d'un tel modèle est son coût, à la fois de représentation et de manipulation. Pour résumer, il s'agit de représenter une fonction sphérique quelconque. Cela permet dans un modèle d'éclairement de quantifier le rapport entre énergie incidente et énergie sortante au point d'impact d'un rayon, d'après les propriétés optiques du matériau. Si le modèle diffus idéal, isotrope, se réduit à une BRDF uniforme, le cas plus général fait apparaître des asymétries selon l'angle d'impact. Un miroir, une surface humide, du velours sont autant d'exemples où le cas diffus idéal ne s'applique pas.

L'équation de radiosité étudiée dans les chapitres précédents utilise classiquement le cas diffus idéal, qui se traduit par le facteur $\rho_d(x)$ à l'extérieur de l'intégrale. Il s'agit justement du fait de considérer l'émission *globale* d'un point plutôt que son émission *direction par direction*, qui avait permis de simplifier l'équation d'illumination globale en équation de radiosité, pour rendre sa résolution plus abordable. Modèle BRDF et radiosité sont donc par nature incompatibles puisque cette dernière suppose que l'émission d'un point est isotrope. On le constate d'ailleurs aisément en considérant l'équation de radiosité sous forme de système linéaire utilisant la matrice des facteurs de forme : il ne s'y trouve aucune notion de direction permettant d'injecter une BRDF.

Remonter à l'équation d'illumination globale impose d'utiliser des algorithmes adaptés, principalement basés sur le lancer de rayon. Or, la méthode de radiosité basée sur des voxels, que nous avons présentée dans le chapitre précédent, fait un usage explicite des rayons. Il est donc relativement aisé d'y introduire le modèle BRDF. L'implémentation de ce modèle est une thématique de recherche connue [Neu01, Rus97, Rus98]. La technique la plus répandue utilise une décomposition sur la base de fonctions des *harmoniques*

*sphériques*, très adaptées à la représentation d'une fonction sphérique (cf. section 5.1.4 page 122). Pourtant, dans le cadre de la synthèse d'image, nous avons pensé qu'un modèle plus simple pouvait être utilisé, notamment pour les BRDF de formes assez lisses. En effet, la représentation des fonctions n'est pas le seul point à considérer, il existe également une difficulté classique dans le calcul des rotations. Les rotations sont nécessaires dans un calcul d'éclairement, mais grèvent les performances des harmoniques sphériques. Nous avons donc mis au point une méthode utilisant une interpolation entre points de contrôle, qui s'avère effectivement largement plus efficace en temps de calcul.

Ce chapitre présente tout d'abord la problématique de l'utilisation des BRDF, ainsi que l'outil classique des harmoniques sphériques. Ensuite, nous détaillons la construction de notre modèle plus simple, dont nous comparons l'efficacité avec le précédent. Enfin, nous montrons comment introduire les BRDFs dans notre algorithme de radiosité.

## 5.1 Utilisation d'une BRDF

### 5.1.1 Représentation

Une BRDF est une fonction de deux paramètres : la direction de l'énergie incidente, et la direction du rayon sortant dont on veut connaître la proportion de ré-émission. Chaque direction est en réalité un paramètre à deux dimensions, généralement découplé en deux angles polaires $\theta$ et $\phi$, représentant l'élévation et l'azimut (Figure 5.1). Une BRDF est donc plutôt vue comme fonction de quatre paramètres $(\theta_{in}, \phi_{in}, \theta_{out}, \phi_{out})$. Les valeurs prises par cette fonction sont le ratio entre la radiance réfléchie (ou transmise) dans la direction de sortie, et l'énergie reçue dans la direction incidente. La BRDF d'une surface de diffusion idéale est une constante, tandis qu'un miroir parfait répond par un pic de Dirac. Une BRDF simple peut être représentée par une fonction mathématique, mais une BRDF quelconque nécessite en général un ensemble de valeurs échantillonnées à interpoler, acquises sur un matériau réel. De nombreuses représentations de BRDFs ont été proposées dans la littérature [War92, KM99, AS00], la plupart du temps basées sur une décomposition sur une base de fonctions, comme les harmoniques sphériques ou les ondelettes sphériques [SS95, KvDS96, CPB06].

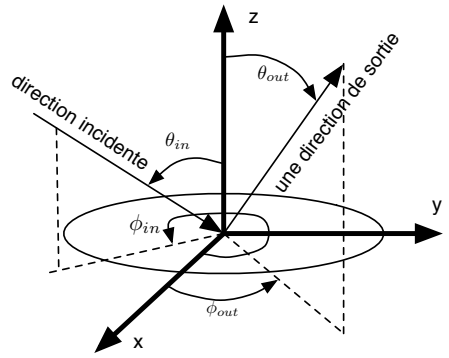



FIG. 5.1 – Une BRDF est paramétrée par des angles polaires : $\theta$ est l'angle zénithal caractérisant l'élévation, $\phi$ est l'angle azimutal caractérisant l'orientation.

### 5.1.2 Modèles et instanciations : BRDF et distributions de radiance

En déclarant que l'on utilise une BRDF dans le modèle d'éclairement, on ne fait que décrire une certaine propriété du matériau. Il ne faut cependant pas perdre de vue que cette propriété est *statique* et diffère de l'*émission effective* de chaque élément constituant un morceau de surface dans la scène illuminée. Considérons en effet une scène 3D dont nous calculons l'illumi-

nation globale. Elle est habituellement subdivisée en patches (éléments surfaciques), chacun se comportant comme une source de lumière. Les patches de la même surface partagent la même BRDF, qui décrit leur *comportement* caractéristique du matériau sous-jacent. En revanche, l'émission effective de chaque patch diffère de celle du patch voisin, en fonction justement de l'énergie reçue et de l'énergie ré-émise, calculées par l'algorithme d'illumination globale. La définition exacte de l'« émission effective » est une *distribution de radiance*, qui n'est fonction que d'un paramètre : la direction de sortie considérée (Figure 5.2). Ici encore, on peut la voir comme fonction plutôt des deux paramètres $\theta$ et $\phi$. Cette fonction n'est pas connue *a priori*, elle résulte en tout point du calcul d'illumination globale. Initialement, seules les sources lumineuses ont des distributions de radiance non nulles ; ensuite, au fur et à mesure que la lumière se propage et heurte des patches, cela affecte les distributions de radiance de ces derniers en fonction de leurs BRDFs respectives. Résoudre le problème de l'illumination globale signifie que l'on connaît les BRDFs (fonctions de 4 paramètres) et que les inconnues sont les distributions de radiance (fonctions de 2 paramètres).

À des fins d'optimisation, ces deux objets différents sont souvent représentés par les mêmes structures de données. Plutôt que d'encoder une fonction à 4 paramètres, une BRDF peut être vue comme l'interpolation de plusieurs fonctions à 2 paramètres. En d'autres termes, $f(\theta_{in}, \phi_{in}, \theta_{out}, \phi_{out})$ est calculée en interpolant des fonctions $g_{\theta_{in},\phi_{in}}(\theta_{out}, \phi_{out})$ ; connaissant les valeurs de plusieurs $g_{\theta_{in_i},\phi_{in_i}}(\theta_{out}, \phi_{out})$, on peut interpoler ces valeurs, pour un couple $(\theta_{in}, \phi_{in})$ donné. Dans [cXSAWG91], les BRDFS sont représentées par des harmoniques sphériques, et ces sont les coefficients de ces harmoniques qui sont interpolés pour générer la valeur cherchée. Ainsi, les BRDFs comme les distributions de radiance peuvent être représentées par des fonction sphériques 2D, et une seule technique est nécessaire pour encoder efficacement de telles données.

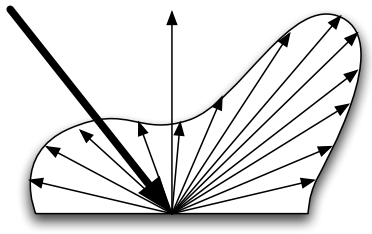


FIG. 5.2 – Une distribution de radiance est l'accumulation de la réalisation de BRDFs pour des incidences données. Celle de cette figure est principalement diffuse avec un comportement spéculaire assez doux.

### 5.1.3 Opérations

Ainsi qu'il est expliqué dans la section précédente, la gestion des distributions de radiance implique un encodage de fonctions sphériques 2D. Leur utilisation passe quant à elle par quelques opérations élémentaires. Ces opérations sont la requête de valeur, la multiplication par un facteur, l'addition, et la rotation des distributions. Ces opérations essentielles doivent être très efficaces.

**Requête de valeur :** La distribution de radiance d'un élément de la scène évolue au fur et à mesure que cet élément reçoit de l'énergie, durant la résolution de l'illumination globale. Il est nécessaire de pouvoir à tout moment connaître l'énergie émanant d'un élément dans une direction donnée. Cela peut être effectué par l'utilisation directe d'un modèle mathématique sous-jacent, ou par une interpolation en fonction d'un certain voisinage. Un tel modèle mathématique est décrit en section 5.1.4 page suivante, tandis que nous proposons en section 5.2 page 124 une approche par interpolation.

**Multiplication par un facteur et addition :** Lorsque de la lumière heurte un élément de la scène, sa BRDF locale est interrogée pour récupérer la distribution de radiance devant résulter de cette interaction. Cette distribution doit alors être mise à l'échelle de l'énergie reçue, et ajoutée à la distribution déjà présente dans l'élément. La figure 5.3 illustre ces opérations.

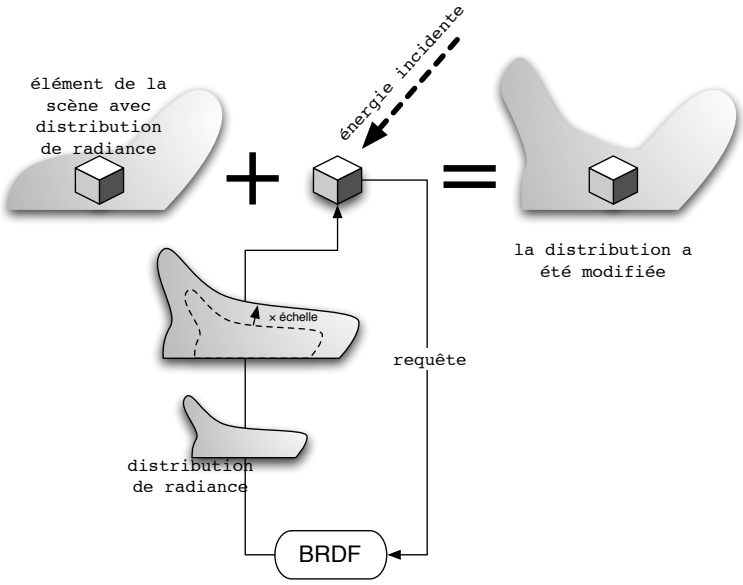



FIG. 5.3 – Multiplication et addition de distributions de radiance

**Rotation :** Les distributions de radiance renvoyées par des requêtes auprès des BRDFs sont calculées dans un repère global. Il faut leur faire subir une rotation pour les orienter dans le repère local de l'élément à modifier. En effet, l'angle zénithal de la distribution correspond à l'angle formé avec le vecteur normal à la surface au point considéré, et l'angle azimutal est pertinent si le matériau est anisotrope, comme une fourrure lissée.

### 5.1.4 Une solution classique : les harmoniques sphériques

Une solution classique pour répondre aux différentes attentes précédemment évoquées est la décomposition des distributions de radiance sur la base de fonctions que constituent les *harmoniques sphériques*. Cette base étant connue, la distribution est ainsi représentée par un simple jeu de coefficients. Ajouter et multiplier des distributions s'effectue en ajoutant et multipliant les coefficients. En revanche, la rotation est généralement la pierre d'achoppement.

**Les harmoniques sphériques :** Les harmoniques sphériques constituent une base de fonctions particulièrement bien adaptée à la représentation de fonctions sphériques 2D [FC94, KSS02]. Les fonctions de la base sont notées $Y_{l,m}$, et une fonction $f$ est décomposée avec des coefficients $C_{l,m}$ :

$$f(\theta,\phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} C_{l,m} Y_{l,m}(\theta,\phi)$$

La définition des fonctions $Y_{l,m}$ utilise les polynômes de Legendre associés $P_{l,m}(x)$ et une fonction de normalisation $N_{l,m}(x)$.

$$Y_{l,m}(\theta,\phi) = \begin{cases} N_{l,m} P_{l,m}(\cos\theta)\cos(m\phi) & \text{si } m > 0 \\ N_{1,0} P_{1,0}(\cos\theta)\sqrt{2} & \text{si m} = 0 \\ N_{l,m} P_{l,-m}(\cos\theta)\sin(-m\phi) & \text{si } m < 0 \end{cases}$$

$$N_{l,m} = \sqrt{\frac{2l+1}{2\pi}\frac{(l-|m|)!}{(l+|m|)!}}$$

$$P_{l,m}(x) = \begin{cases} (1-2m)\sqrt{1-x^2}P_{m-1,m-1}(x) & \text{si } l = m \\ x(2m+1)P_{m,m} & \text{si } l = m+1 \\ x\frac{2l-1}{l-m}P_{l-1,m}(x) - \frac{l+m-1}{l-m}P_{l-2,m}(x) & \text{sinon} \end{cases}$$

**Coût de représentation :** La représentation par harmoniques sphériques impose le choix du nombre de coefficients à conserver dans la décomposition. Les BRDFs considérées étant généralement lisses, quelques dizaines de coefficients suffisent. Il est important de choisir le nombre le plus faible

possible de coefficients, car les distributions étant présentes dans chaque élément de la scène, la mémoire devient vite une ressource critique. Un bon compromis doit être trouvé entre la précision voulue et le nombre de coefficients conservés (cf. figures 5.5 et 5.6). Pour une fonction réelle, seuls les coefficients réels des harmoniques sphériques sont nécessaires, les coefficients imaginaires étant nuls.

Les surfaces hautement réfléchissantes sont des exemples typiques infirmant l'hypothèse de BRDFs lisses. Cependant, de telles réflexions ne sont pas gérées par un modèle très précis, mais plutôt par des astuces algorithmiques. De telles astuces sont citées dans [cXSAWG91].

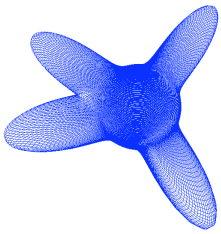


FIG. 5.4 – Une forme sphérique à approcher, faite d'une sphère étirée par endroits.

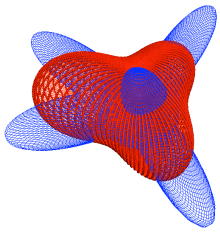


FIG. 5.5 – Harmoniques sphériques, 16 coefficients réels. Le résultat (sphères rouges) est loin d'être suffisant.

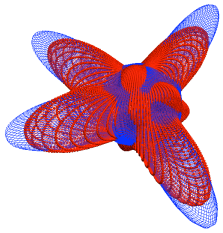


FIG. 5.6 – Harmoniques sphériques, 64 coefficients réels. Le résultat (sphères rouges) est une approximation correcte.

**Rotation avec les harmoniques sphériques :** La rotation avec les harmoniques sphériques est connue comme le goulot d'étranglement des calculs associés à cette représentation [KKB+05]. Même s'il s'agit théoriquement d'une opération linéaire [BFB97, Öhr], correspondant à un calcul matrice-vecteur, la matrice en question est extrêmement difficile à obtenir. Des astuces sont parfois déployées pour simplifier le calcul dans certains cas particuliers [cXSAWG91, KKB+05].

## 5.2 Une solution par points de contrôle

Dans l'objectif d'obtenir les meilleures performances possibles, nous proposons une méthode d'implémentation des BRDFs basée sur une interpolation dans des triangles sphériques. Cette méthode requiert un nombre raisonnable de coefficients, est assez précise pour des distributions de radiance assez lisses, et est bien adaptée à notre algorithme de résolution de l'illumination globale « orienté directions ».

### 5.2.1 Interpolation

La façon la plus simple de représenter une fonction sphérique 2D est de l'échantillonner en certains points. Si les angles polaires $(\theta, \phi)$ de ces points de contrôle sont fixés, la représentation ne nécessite de stocker que la composante $r$ des coordonnées polaires $(r, \theta, \phi)$ de ces points.

Deux difficultés doivent être traitées : choisir l'ensemble des points de contrôle, et choisir un algorithme d'interpolation.

#### 5.2.1.1 Choisir l'ensemble des points de contrôle

Les points de contrôle doivent être répartis uniformément sur la sphère, de telle sorte qu'aucune région ne soit arbitrairement plus précise qu'une autre. Une fois répartis, les points de contrôle sont translatés radialement pour étirer et compresser localement la sphère et lui donner la forme voulue.

Une bonne méthode de répartition des points est la subdivision récursive d'un polyèdre [SS95]. Par exemple, partant d'un icosaèdre (12 sommets), on peut passer à la subdivision suivante en prenant le milieu de chaque arête pour créer dans chaque face triangulaire quatre nouveaux triangles (Figures 5.7 et 5.8).

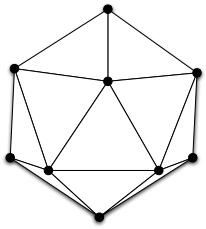


FIG. 5.7 – Icosaèdre, ordre 0 de subdivisions

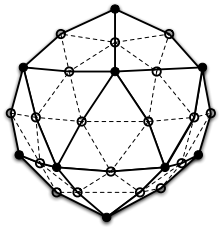


FIG. 5.8 – Partant de l'icosaèdre, ordre 1 de subdivision

Le polyèdre initial est un paramètre important pour déterminer le nombre de points de contrôle. Avec un icosaèdre, le nombre de points augmente rapidement ; aux ordres 0, 1, 2, 3 de subdivisions, ce nombre devient 12, 42, 162, 642. Ainsi qu'il est expliqué en section 5.1.4 page 122, chaque élément

de la scène contient un jeu complet de coefficients, si bien que son cardinal doit être conservé le plus petit possible pour économiser la mémoire, tout en étant assez important pour assurer une bonne précision. Nos expériences ont évalué que 162 points (ordre de subdivision 2 de l'icosaèdre) représentent un bon compromis (Figures 5.9, 5.10 et 5.11). Sur ces illustrations, les points représentés ne sont pas les points de contrôle, mais des échantillons interpolés par notre méthode, en utilisant les points de contrôle.

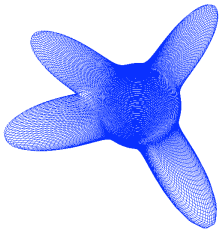


FIG. 5.9 – Une forme sphérique à interpoler, faite d'une sphère étirée par endroits.

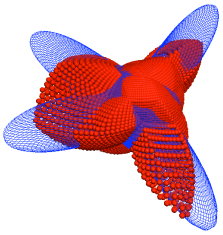


FIG. 5.10 – 42 points de contrôle. Le résultat (sphères rouges) est loin d'être suffisant.

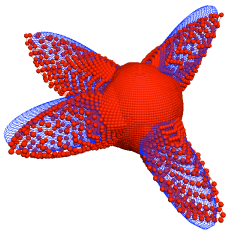


FIG. 5.11 – 162 points de contrôle. Le résultat (sphères rouges) est une approximation correcte.

#### 5.2.1.2 Attribution des poids

La subdivision décrite ci-dessus définit les points de contrôle. Il faut également définir une stratégie d'attribution de poids pour réaliser une interpolation.

Soit un point $P$ à interpoler. Si l'on projette $P$ et les points de contrôle sur une sphère unitaire, alors la projection $P'$ de $P$ tombe dans une face courbe triangulaire formée par trois points de contrôle. Dans certains cas, $P'$ peut se trouver sur une arête ou un sommet, mais cela n'est pas gênant.

Pour une interpolation dans un triangle, l'interpolation bi-linéaire est courante en informatique graphique. Mais cette approche est surtout valable pour des triangles plats et rastérisés (représentés par tramage de points) dans un buffer que l'on peut parcourir ligne par ligne. Ici, nous avons choisi d'utiliser une pondération s'appuyant sur les aires des triangles que l'on peut tracer avec $P'$ et les points de contrôle voisins. Ces triangles sont des triangles sphériques dont on sait calculer l'aire par la formule :

$$aire(ABC) = \widehat{A} + \widehat{B} + \widehat{C} - \pi$$

Soit $P$ le point à interpoler, dont la projection $P'$ sur la sphère unitaire tombe dans le triangle sphérique défini par les projections $C'_1$,$C'_2$,$C'_3$ des points de contrôle $C_1$,$C_2$,$C_3$. L'interpolation est définie par $P = \omega_1 C_1 + \omega_2 C_2 + \omega_3 C_3$, les inconnus étant les poids $\omega_i$.

Ces poids sont calculés d'après les aires des triangles formés par les points projetés. Soit les triangles (éventuellement réduits à un segment ou un point) $P'C'_1C'_2$, $P'C'_1C'_3$ et $P'C'_2C'_3$, d'aires respectives $\alpha_1$, $\alpha_2$ et $\alpha_3$. On peut définir $\omega_i = \frac{\alpha_i}{\alpha_1+\alpha_2+\alpha_3}$ comme système de poids normalisés (cf. figure 5.12).

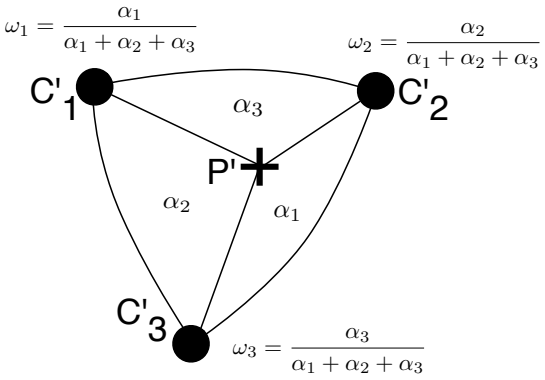



FIG. 5.12 – Interpolation dans un triangle sphérique. Les poids $\omega_i$ sont déterminés d'après les aires $\alpha_i$

#### 5.2.1.3 Détermination du triangle

Pour déterminer quels sont les trois points de contrôle rentrant en jeu dans l'interpolation d'un point $P$ donné, il est nécessaire de calculer rapidement le triangle où tombe la projection de ce dernier. Les coordonnées polaires de $P$ permettent d'identifier assez rapidement quelques triangles candidats : il suffit pour cela de trier les points de contrôle par parallèles et méridiens. Deux triangles sont alors être proposés, et quelques produits vectoriels permettent de déterminer quel est le bon, en fonction de la position de $P'$ par rapport à l'arête commune.

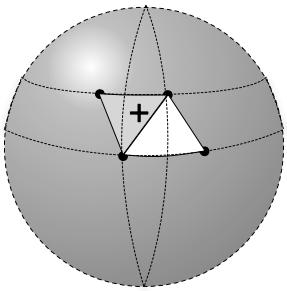


FIG. 5.13 – Les parallèles et méridiens des points de contrôle aident à déterminer des triangles candidats à contenir la projection d'un point donné.

### 5.2.2 Représentation d'une fonction existante

La section 5.1.2 page 119 a établi que même les BRDFs à quatre paramètres peuvent être modélisées par des fonctions sphériques à deux paramètres, ce qui permet d'unifier la représentation des BRDFs et des distributions de radiance. Il est donc nécessaire de proposer un algorithme permettant de trouver un bon ensemble de points de contrôle pour représenter une fonction donnée. Notre algorithme d'interpolation, qui suppose les angles polaires fixés pour les points de contrôle, implique donc que nous cherchions simplement les distances radiales des points de contrôle au centre.

Pour déterminer un tel ensemble, optimal, il est possible de transformer le problème en un système d'équations pouvant être résolu par la méthode des moindres carrés, ou par programmation linéaire.
Soit $f$ une fonction sphérique 2D à représenter, et $P_i(r_i, \theta_i, \phi_i)$, $i \in [0; n]$ des échantillons de la fonction. Soient $C_j(r_j, \theta_j, \phi_j)$, $j \in [0; m]$ les points de contrôle (on a normalement $n > m$). Les inconnues sont les $r_j$.
Notre méthode d'interpolation stipule que $P_i$ est interpolé par au plus trois points de contrôle $C_1$, $C_2$ et $C_3$, de telle sorte que $r_i = \omega_{i,1} r_1 + \omega_{i,2} r_2 + \omega_{i,3} r_3$.
Soit $X$ le vecteur des inconnues $(r_j)_{j \in [0;m]}$, c'est-à-dire les distances radiales des points de contrôle $C_j$. Soit $Y$ le vecteur $(r_i)_{i \in [0;n]}$ des distances radiales des échantillons $P_i$.
Soit $A$ la matrice des poids des interpolations. Chaque ligne de la matrice représente l'interpolation d'un échantillon, et a au plus trois composantes non nulles.
Le système à résoudre est $AX = Y$. Il est usuellement surdéterminé, puisque l'on peut disposer d'autant d'échantillons que désiré, tandis que le nombre de points de contrôle reste fixe. Il n'a vraisemblablement pas de solution ; c'est donc une pseudo-solution, optimale d'après certains critères, qui doit être calculée.

#### 5.2.2.1 Méthode des moindres carrés

Pour résoudre le système $AX = Y$, la méthode des moindres carrés est une méthode de choix, qui choisit $X = (A^t A)^{-1} Y$ comme sa meilleure solution. En pratique, nous avons obtenu d'excellents résultats avec cette technique sur des fonctions lisses.

#### 5.2.2.2 Programmation linéaire

L'inconvénient majeur de la méthode des moindres carrés est qu'elle autorise une erreur locale assez grande pour peu que l'erreur reste faible partout ailleurs. Marzais *et al* proposent une approche différente [MGM06], par programmation linéaire, pour éviter cet écueil. En minimisant la norme infinie de l'erreur, cette méthode essaye de limiter toutes les erreurs locales.

Pour notre problème, cette technique de programmation linéaire consiste à résoudre l'équation (5.1) de la présente page.

$$-\mathbb{1}.h \leq ((AX - Y)_i)_{i\in[0;n]} \leq \mathbb{1}.h \tag{5.1}$$

$h$ est l'erreur à minimiser

Cependant, de meilleurs résultats peuvent être obtenus dans notre cas en minimisant l'erreur *relative* (Équation 5.2). Cela permet aux zones de faibles coefficients d'être moins perturbées, en sacrifiant un peu de précision pour les zones de grands coefficients.

$$-\mathbb{1}.h \leq \left(\frac{(AX - Y)_i}{Y_i}\right)_{i\in[0;n]} \leq \mathbb{1}.h \tag{5.2}$$

Un compromis entre les deux solutions précédentes peut être obtenu en divisant l'erreur par la racine de la valeur (Equation 5.3). Il s'agit alors d'une erreur relative modifiée.

$$-\mathbb{1}.h \leq \left(\frac{(AX - Y)_i}{\sqrt{Y_i}}\right)_{i\in[0;n]} \leq \mathbb{1}.h \tag{5.3}$$

La figure 5.11 page 125 montre que certaines zones sont très bien approchées, tandis que d'autres sont moins précises. Effectivement, si l'erreur maximale se situe au niveau des pics, il peut s'agir d'une faible erreur à l'échelle du pic, mais d'une erreur très grossière ailleurs. En d'autres termes, si l'erreur ne peut être inférieure à $\epsilon$, cela peut causer des perturbations dans l'intervalle $[-\epsilon; \epsilon]$ risquant de gâcher les zones d'ordre de grandeur $\epsilon$. En choisissant plutôt une erreur relative, chaque zone est contrainte à la plus petite erreur par rapport à son échelle. Les figures 5.14, 5.15 et 5.16 illustrent cela avec une forme constituée d'un pic assez large.

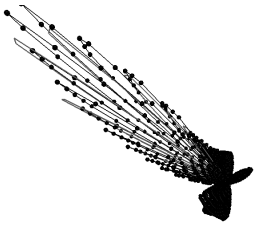


FIG. 5.14 – La minimisation de l'erreur absolue provoque de fortes perturbations des zones à faibles coefficients.

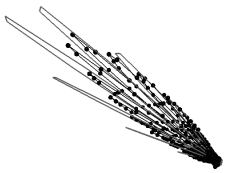


FIG. 5.15 – La minimisation de l'erreur relative améliore les zones à faibles coefficients mais dégrade les zones fortement valuées.

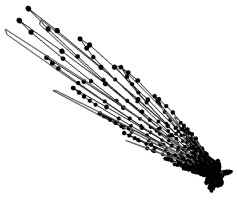


FIG. 5.16 – Un compromis peut être trouvé en divisant par la racine de la valeur plutôt que par la valeur elle-même.

Après de nombreux tests, la méthode des moindres carrés s'est cependant toujours révélée un peu meilleure que la méthode par programmation linéaire. Ces deux méthodes sont simples à implémenter et peuvent présenter un intérêt pour tous types de formes.

### 5.2.3 Déformation locale des représentations existantes

L'ensemble des coefficients caractérisant la décomposition d'une fonction, sur une base de fonctions, ne peut pas toujours être facilement interprété. Si pour les B-Splines ou les NURBS on dispose d'un contrôle local, les harmoniques sphériques n'ont pas un tel avantage. Les premiers coefficients façonnent sommairement la forme, les suivants apportant des détails ; mais une perturbation de la forme a vraisemblablement un impact sur tous les coefficients. Aussi, il est difficile de déformer intuitivement une fonction en agissant simplement sur les coefficients, car les conséquences sont difficiles à prévoir.

Notre méthode d'interpolation présente la particularité que chaque point est caractérisé par trois coefficients au plus, qui peuvent être aisément déterminés. Éloigner ou rapprocher un point du centre n'a donc d'effet que sur ces trois coefficients. Pour changer la distance radiale d'un point, il suffit donc de recalculer les poids des trois points de contrôle associés.

Soit l'interpolation de $P(r, \theta, \phi)$ par les points de contrôle $C_1(r_1, \theta_1, \phi_1)$, $C_2(r_2, \theta_2, \phi_2)$ et $C_3(r_3, \theta_3, \phi_3)$ correspondant à $r = \omega_1 r_1 + \omega_2 r_2 + \omega_3 r_3$.
Soit $P'(r + \delta r, \theta, \phi)$ une perturbation de $P$. Il faut donc trouver $\delta r_1$, $\delta r_2$ et $\delta r_3$ tels que $r + \delta_r = \omega_1(r_1 + \delta r_1) + \omega_2(r_2 + \delta r_2) + \omega_3(r_3 + \delta r_3)$.
Une solution triviale mais non satisfaisante est $\delta r = \delta r_1 = \delta r_2 = \delta r_3$, puisque

$$\omega_1(r_1+\delta r)+\omega_2(r_2+\delta r)+\omega_3(r_3+\delta r) = \underbrace{\omega_1 r_1 + \omega_2 r_2 + \omega_3 r_3}_{r} + \delta r \underbrace{(\omega_1 + \omega_2 + \omega_3)}_{1}$$

Cette solution n'est pas satisfaisante car elle ne fait que décaler uniformément le triangle sphérique dans lequel tombe la projection de $P$. Cela risque de perturber trop fortement tous les points interpolés dans les triangles contenant $C_1$, $C_2$ ou $C_3$. Il est plus intéressant de moduler les $\delta r_i$ par les $\omega_i$, pour décaler les points de contrôle conformément à leur importance dans l'interpolation de $P$.

Si l'on impose par exemple $\delta r_i$ proportionnel à $\omega_i \delta r$, il suffit de diviser par $\sum \omega_j^2$ pour normaliser les poids. La solution que nous avons choisie est donc la suivante :

$$\delta r_i = \frac{\omega_i}{\sum_j \omega_j^2} \delta r$$

### 5.2.4 Rotation

Notre méthode d'interpolation, pour être efficace, fixe les angles polaires. Ceux-ci ne peuvent être modifiés. La rotation ne peut donc pas être effectuée en appliquant une simple rotation aux points de contrôle. Il est possible, mais peu raisonnable, de recalculer tous les points de contrôle en rééchantillonnant la fonction à laquelle on fait subir la rotation, et en réappliquant la méthode des moindres carrés. La rotation étant une fonction critique, elle devrait être exécutée le plus rapidement possible.

#### 5.2.4.1 Approche par permutations

Dans notre système de points de contrôle, une rotation correspond approximativement à une permutation des points de contrôle. Il s'agit d'associer à chacun, *virtuellement modifié par la rotation*, un autre des points de contrôle, réel ; vraisemblablement le plus proche (figure 5.17 page ci-contre).

Un tel processus est très rapide puisqu'une rotation peut être associée à une permutation, parmi plusieurs permutations précalculées à différents angles jugés pertinents. L'inconvénient est que la rotation n'est alors plus une opération continue, qu'elle peut devenir l'identité pour des angles trop petits, et qu'elle peut accumuler de l'erreur facilement dans une succession d'applications.

Pour calculer la permutation correspondant à une rotation, une méthode naïve serait d'apparier chaque point virtuellement tourné au point de contrôle réel le plus proche, en partant du meilleur appariement (plus petite erreur) et en terminant par le pire. Le problème survient assez vite au fur et à mesure des appariements : ceux-ci peuvent devenir catastrophiques au fur et à mesure que de moins en moins de points de contrôle restent disponibles (figure 5.18 page suivante).

Pour surmonter ce problème, on peut adopter une autre stratégie : d'abord réaliser le meilleur appariement (plus petite erreur) qui a lieu, disons avec le point $C$. Ensuite, les appariements sont effectués au mieux, mais en partant du point le plus proche de $C$ et terminant par le plus éloigné. Cela permet d'éviter que tous les voisins d'un point réalisent leur appariement avec les alentours, laissant ce point isolé, cause des appariements catastrophiques. La figure 5.19 page ci-contre est une illustration d'ordre de parcours des points à apparier.

#### 5.2.4.2 Approche continue

La meilleure représentation possible d'une forme ayant subi une rotation pourrait être trouvée en effectuant à nouveau un échantillonnage, et le calcul des moindres carrés (cf. section 5.2.2 page 127) sur les images des échantillons après rotation. Le coût en est prohibitif, mais cela souligne bien

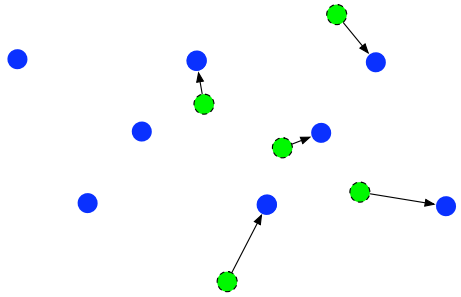


FIG. 5.17 – Chaque point de contrôle virtuellement modifié par rotation (vert, bord en pointillés) est associé au plus proche point de contrôle, réel (bleu, bord plein).

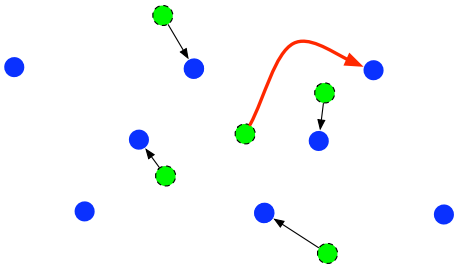


FIG. 5.18 – Partir du meilleur appariement et terminer par le pire peut provoquer des appariements catastrophiques au fur et à mesure que les points ne sont plus disponibles. Le point central est ici assez mal apparié.

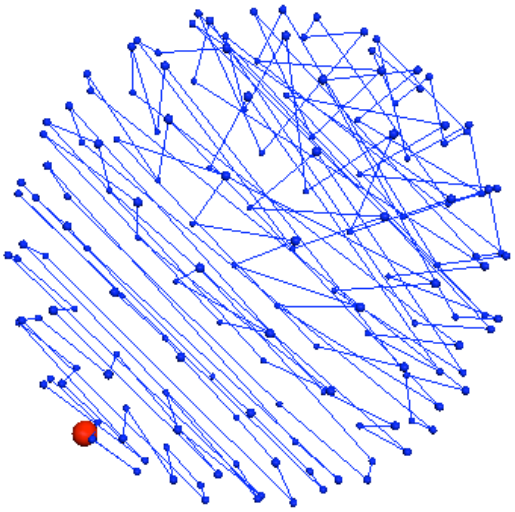


FIG. 5.19 – Le meilleur appariement est réalisé au point en bas à gauche. Les suivants sont réalisés en s'en éloignant.

qu'il existe un jeu de coefficients optimal à trouver, qui peut être assez différent d'une simple permutation. En réalité, il est possible de s'en approcher, en effectuant une interpolation, mais cette fois en interpolant non pas les échantillons, mais *les points de contrôle* eux-mêmes, considérés après rotation (Figure 5.20 page suivante).

Chaque point de contrôle, s'il subit une rotation, peut être considéré comme un échantillon, interpolable parmi trois points de contrôle réels. Un point de contrôle réel peut donc être impliqué dans l'interpolation de plu-

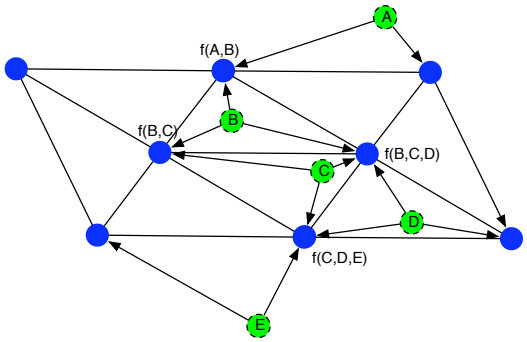



FIG. 5.20 – Les points de contrôle réels (bleu, trait plein) interpolent les images de ces mêmes points par rotation (vert, trait en pointillés). Ils ne sont pas impliqués dans les mêmes interpolations. Une pondération $f$ des points qu'ils interpolent doit permettre de leur donner une nouvelle distance radiale convenable.

sieurs de ses homologues virtuellement tournés. On lui associe ainsi un poids par interpolation. Et au final, on fixe sa distance radiale grâce à une somme pondérée des interpolations auxquelles il participe. Sa nouvelle position radiale ne satisfait généralement pas exactement les interpolations qu'il doit réaliser, mais ce processus simplifié de rotation redevient ainsi un processus continu.

Tout comme avec les permutations, il est possible de précalculer un certain ensemble de ces rotations, pour des angles jugés pertinents, car elles sont facilement représentables avec un simple jeu de coefficients.

### 5.2.5 Résolution de l'illumination globale favorisant les directions

L'algorithme de radiosité sur des voxels que nous avons présenté au chapitre 4 utilise une méthode particulière du calcul de la visibilité entre éléments volumiques. L'algorithme est paramétré par un ensemble de directions échantillonnant les rayons pouvant être considérés. La lumière n'est transportée que dans ces directions. Les calculs de visibilité sont factorisés sur les rayons intersectant la scène (figure 5.21 page suivante).

Une conséquence que l'on peut tirer ce cet algorithme favorisant les directions, est qu'il est aussi possible de factoriser les calculs d'échanges d'énergie effectués sur un rayon. En effet, la méthode d'interpolation par points de contrôle que nous avons décrite repose sur le calcul de trois poids pour une direction $(\theta, \phi)$ donnée. Combinée avec notre algorithme de radiosité, ce calcul de poids peut ainsi être factorisé pour tous les éléments intersectant les rayons dans une direction donnée. C'est une optimisation supplémentaire que permet notre algorithme, et qui est loin d'être négligeable. Dans

certains cas, elle est quasiment indispensable, notamment pour les calculs de transfert radiatifs que nous avons expérimentés, présentés au chapitre 6, section 6.3.2 page 149.

Dans cette expérimentation, nous avons considéré les échanges d'énergie au sein d'un nuage. La propagation de l'énergie est appréhendée par des *fonctions de phase* [Len93]. Ces fonctions de phase établissent la probabilité pour un photon de continuer sa route tout droit ou d'être dérouté dans une autre direction, et elles peuvent être représentées par des fonctions sphériques. Dans le cas du nuage, les voxels ne sont pas orientés par des normales ; les fonctions de phase sont alignées sur la direction courante, mais ne subissent pas de rotation locale à chaque voxel. Les distributions de « radiance » encodées par des fonctions sphériques sont présentées dans un repère global. Chacun des $N$ voxels du nuage est interrogé sur la quantité d'énergie qu'il ré-émet dans la direction courante. Le calcul des poids effectué pour réaliser l'interpolation dans cette direction est donc valide pour tous les voxels. On divise donc le coût associé à l'opération de « requête de valeur » par $N$.

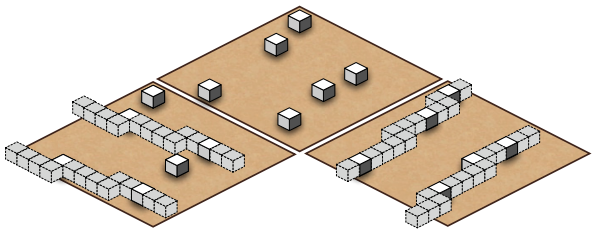


FIG. 5.21 – Un algorithme favorisant les directions peut factoriser le calcul des poids lors de l'interpolation des distributions de radiance, pour tous les éléments appartenant aux rayons.

## 5.3 Résultats expérimentaux

### 5.3.1 Comparaison avec un calcul par harmoniques sphériques

Pour comparer notre implémentation des BRDFs utilisant des points de contrôle, avec une implémentation basée sur les harmoniques sphériques, nous avons utilisé notre algorithme de radiosité. Les harmoniques sphériques reposent sur le SpharmonicKit [KR], qui a d'excellentes performances. Un écart de performances constaté entre notre algorithme d'interpolation et les harmoniques sphériques n'est donc pas dû à une implémentation naïve de ces dernières.

La figure 5.22 montre que notre méthode est largement plus performante que l'approche par harmoniques sphériques. De plus, nous n'avons pas implémenté la rotation pour ces dernières, ce qui fait qu'elles ne sont pas même ralenties par leur goulet d'étranglement connu. La scène utilisée est un simple assemblage de cubes, utilisant des BRDFs diffuses. Elle a été discrétisée en 13 000 voxels pour l'algorithme de radiosité (Figure 5.23 page ci-contre).

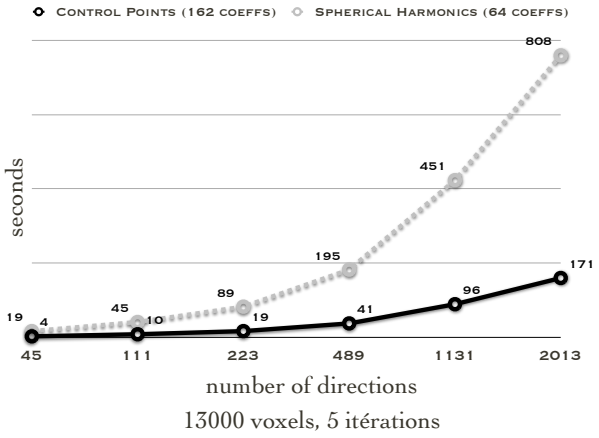



Fig. 5.22 – Les performances de la méthode par points de contrôle (trait plein) dépassent largement celles des harmoniques sphériques (pointillés).

### 5.3.2 Comparaison avec l'algorithme de radiosité sans BRDF

L'insertion de BRDFs dans notre algorithme de radiosité est assez coûteux en espace, mais ne change pas la complexité théorique de résolution. En revanche, le temps de traitement effectif est malgré tout augmenté par les opérations appliquées aux BRDF (requête, rotation...). Nous avons cependant observé que notre implémentation est assez efficace comparée au temps de calcul du même algorithme en radiosité simple, c'est-à-dire sans BRDFs.

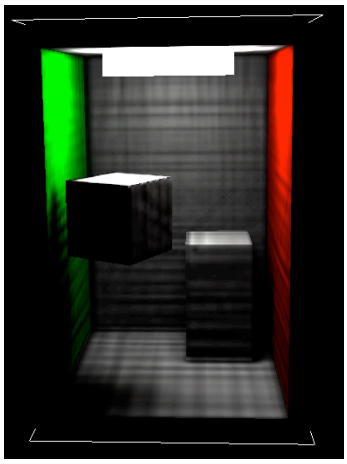


FIG. 5.23 – La scène utilisée pour les expérimentations. L'effet d'aliasing est dû au faible nombre de directions et aux capacités limitées du visualiseur utilisé.

Pour une scène test d'environ 370 000 voxels, nous avons par exemple observé les temps de calcul suivants :

| | |
|---|---|
| Radiosité simple | 1m14s |
| Utilisation de BRDFs | 3m20s |

Le temps de calcul observé pour l'algorithme avec BRDF est environ trois fois supérieur, ce qui est assez bon, dans la mesure où l'on passe de 1 coefficient ($\rho_d(x)$) à 162 coefficients (les points de contrôle). Ce facteur varie sans doute avec les BRDFs utilisées.

Les temps observés sont stables d'une itération sur l'autre, sauf pour la première itération. Il semble en effet que la première itération, qui propage beaucoup de 0, soit plus rapide à l'exécution grâce à des traitements particuliers de la FPU[1] dans le cas de valeurs nulles. Les temps donnés plus haut sont donc ceux de la deuxième itération.

Cette particularité de la première itération n'est pas observable pour l'algorithme sans BRDFs. Il est probable que cela soit dû au fait que le code généré pour les BRDFs et les tableaux de coefficients mette en jeu des instructions vectorielles, et que c'est sur ces instructions que la différence sera notable dans le cas de valeurs nulles.

[1] *Floating Point Unit* unité de traitements des flottants du processeur

## 5.4 Conclusion

La méthode que nous proposons pour représenter des fonctions sphériques 2D implémente efficacement les différents opérateurs requis pour une utilisation des BRDFs dans un algorithme d'illumination globale : addition, multiplication, rotation, interpolation. L'hypothèse des distributions de radiances lisses est importante, car les points de contrôle manquent de précision pour les pics étroits. Le cas des réflexions quasi parfaites peut être traité algorithmiquement plutôt que par le modèle proposé.

Les performances observées dépassent largement celles que l'on peut obtenir par l'utilisation des harmoniques sphériques, pour les même données. À l'usage, plus de mémoire (i.e. de coefficients) est nécessaire, pour obtenir la même précision que les harmoniques sphériques, mais cette quantité reste raisonnable. La mémoire est le seul paramètre limitant pour augmenter la précision.

Cette méthode est bien adaptée à notre solveur de radiosité, qui gagne ainsi un traitement des BRDFs le transformant en solveur d'illumination globale. Elle peut aussi être utilisée efficacement pour des échanges d'énergie volumiques plus complexes comme au sein des modèles de nuages météorologiques (cf. section 6.3.2 page 149).

# Chapitre 6

# Extensions de l'algorithme

L'algorithme de radiosité présenté au chapitre 4 l'a été dans sa forme la plus simple. Il dispose d'une capacité d'évolution sur différents points. D'abord, la forme de l'algorithme se prête bien à la parallélisation, permettant d'envisager efficacement le traitement de données plus lourdes. Ensuite, il est possible d'introduire quelques extensions au modèle d'illumination, notamment pour ce qui est du milieu participatif. Enfin, l'algorithme se présentant comme une brique de calcul relativement indépendante de son environnement, il s'insère dans un projet logiciel modulaire assez souple pour tester différents algorithmes.

## 6.1 Optimisations

### 6.1.1 Modèle par voxels et modèle surfacique

La section 4.5 page 114 révèle quelques optimisations que peuvent offrir les algorithmes de radiosité par patches, contrairement à l'algorithme par voxels, comme la subdivision hiérarchique. Rappelons que cela est principalement dû au fait d'appliquer un algorithme intrinsèquement volumique sur des données mieux représentées par des surfaces. Il est pourtant possible de jouer sur une ambivalence en utilisant les deux modèles en même temps.

#### 6.1.1.1 Partage des distributions de radiance

En maintenant en mémoire une subdivision en patches et une subdivision en voxels, il est possible de connaître tous les voxels appartenant au même patch.

Une première conséquence de cela est qu'il est possible, sous la condition que tous les voxels du patch partagent le même matériau, de factoriser les distributions de radiance de chaque voxel en une seule distribution attribuée au patch. Dans un premier temps, on observe ainsi une économie non négligeable de la mémoire. Dans un deuxième temps, cela permet également

d'économiser des directions. En effet, si les voxels partagent la même distribution de radiance, il n'est plus nécessaire pour un rayon de rencontrer un voxel pour lui propager de l'énergie ; une rencontre avec un voxel quelconque du patch suffit, le patch redistribuant l'énergie à tous ses constituants. L'algorithme voxelique de radiosité réduit alors son rôle à un calcul optimisé des visibilités, mais sa granularité se limite alors aux patches, qui, plus grands, nécessitent certainement moins de directions dans la phase de résolution.

#### 6.1.1.2 Subdivision hiérarchique

En utilisant l'astuce du partage des distributions de radiance entre voxels du même patch, il est possible de bénéficier de la subdivision hiérarchique. En effet, si un patch est subdivisé lorsqu'un critère de trop fort contraste est obtenu, il est trivial de répercuter cette modification sur le modèle utilisant des voxels, en répartissant ces derniers en groupes correspondants aux nouvelles subdivisions du patch.

## 6.2 Parallélisation

La parallélisation d'un algorithme est une optimisation importante. En arrivant à structurer l'algorithme en tâches pouvant être exécutées de façon concurrente, il est possible, d'une part, d'effectuer un calcul plus rapidement, mais également d'envisager de traiter des données plus lourdes en répartissant une charge difficile à appréhender autrement. La parallélisation peut exister à plusieurs niveaux :

1 – utilisation d'instructions particulières du processeur spécialisées dans le traitement vectoriel (SIMD[1], MIMD[2]).

2 – utilisation des threads sur une machine : le programme utilise différents fils d'exécutions concurrents ; il importe pour cela de disposer d'une architecture pouvant effectivement exécuter plusieurs threads en même temps (processeur multithreadé, processeur multicore, système multiprocesseurs)

3 – utilisation de plusieurs machines communiquant par un réseau.

L'optimisation des instructions processeur ne concernant aucune partie spécifique de l'algorithme, il paraît difficile d'apporter quelque chose que ne puisse faire le compilateur. Le calcul des $(i, j)$ de chaque voxel pourrait s'effectuer de façon SIMD, mais il faudrait pour cela réorganiser les données, le jeu n'en valant certainement pas la chandelle.

L'optimisation par threads touche cependant deux points détaillés dans la suite. La premier point, assez facile à implémenter, concerne l'étape de *propagation* de l'algorithme, tandis que le deuxième point, plus ardu, concerne la *répartition* des voxels dans les listes.

L'utilisation d'un réseau de machines est un point essentiel. Notre algorithme de radiosité se prête assez bien à une répartition des calculs et des données, mais les communications deviennent alors le goulet d'étranglement ; il convient dans ce cas de mettre au point des stratégies de répartition des données et d'équilibrage de charge.

L'algorithme a été conçu avec les prémisses d'une utilisation sur grille de calcul ; les travaux approfondis de mise au point des différentes parallélisations sont cependant menés principalement par Rita Zrour [ZCFM06].

### 6.2.1 Distribution des tâches

La distribution des tâches consiste à séparer le problème en actions pouvant être exécutées de façon concurrente, sur les mêmes données. Dans le cas des threads, les fils d'exécutions utilisent la même zone mémoire. Dans le

[1] Single Instruction Multiple Data : application simultanée de la même instruction à différentes données

[2] Multiple Instruction Multiple Data : application simultanée de différentes instructions à différentes données

cas d'un réseau de machines, soit il existe un système de mémoire distribuée, soit les données sont clonées entre les machines (Figures 6.1 et 6.2)

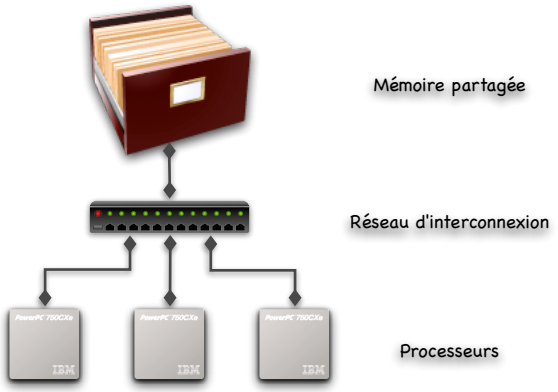



FIG. 6.1 – Mémoire partagée sur un réseau. Les accès à une telle mémoire sont pratiques mais forcément plus coûteux qu'à une mémoire locale. Son utilisation doit être parcimonieuse et judicieuse.

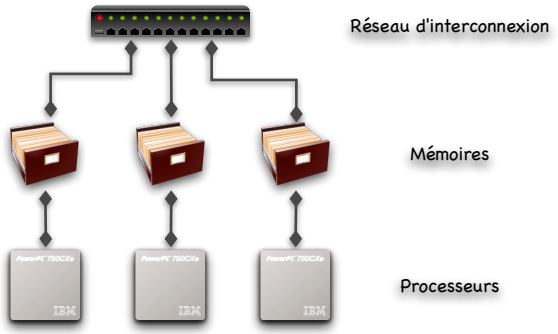



FIG. 6.2 – Mémoire non partagée, les machines n'ont accès qu'à leur mémoire propre. Les données peuvent y être réparties ou dupliquées. Des synchronisations sont souvent nécessaires.

#### 6.2.1.1 Propagation et répartition : threads

**Propagation :** La propagation est l'étape la plus simple à paralléliser. Les voxels ont été répartis en listes, formant une partition. Il s'agit alors de parcourir une fois chaque liste pour transférer de l'énergie entre voxels voisins. Il est possible de répartir les listes en groupes et d'associer un groupe à chaque thread. Les listes ne partageant aucun voxel, il n'y a pas d'effets indésirables dus aux accès concurrents à la même mémoire.
**Répartition :** La répartition des voxels dans les listes est également parallélisable, mais nécessite quelques précautions.
Chaque thread peut traiter une partie des voxels. Cependant, rappelons que ceux-ci sont parcourus dans l'ordre lexicographique pour remplir les listes dans le bon ordre, sans nécessiter de tri supplémentaire. Si plusieurs threads écrivent dans les même listes, l'ordonnancement des threads risque donc de perdre complètement le tri naturel des listes (Figure 6.3). Ce problème peut être contourné en dupliquant les listes. Chaque thread construit donc un segment de chaque liste. Une concaténation des listes homologues de chaque thread constitue alors la liste totale souhaitée (Figure 6.4).
À des fins d'optimisation, cette phase de concaténation est évitée en déportant le « saut » de liste en liste à l'étape de propagation (Figure 6.5 page suivante).

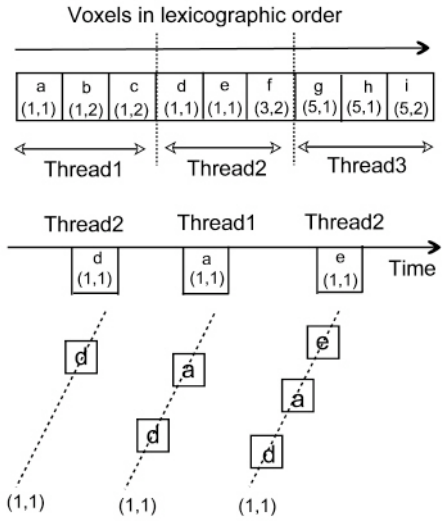



Fig. 6.3 – Répartition en parallèle dans les mêmes listes. L'ordonnancement des threads risque de perdre complètement l'ordre naturel de construction des listes.

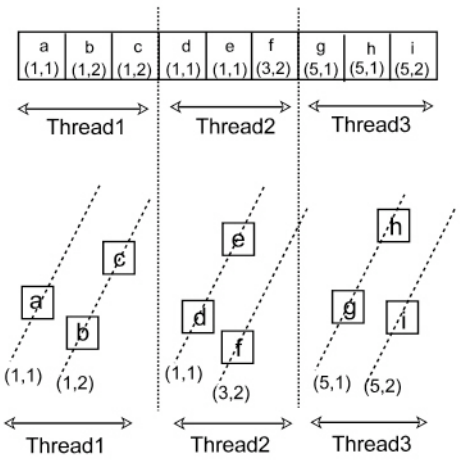



Fig. 6.4 – Répartition en parallèle dans des listes différentes. Chaque thread produit des listes triées qu'il suffit ensuite de concaténer avec leurs homologues respectifs.

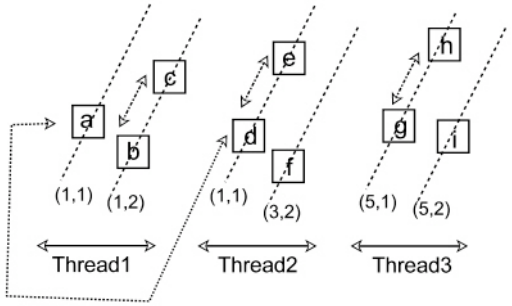



FIG. 6.5 – La propagation s'effectue de sous-liste en sous-liste, sans recréer une nouvelle structure de listes concaténées.

#### 6.2.1.2 Sous-ensembles de directions : réseau de machines

Les threads permettent de paralléliser les deux boucles internes de l'algorithme, sur les voxels et sur les listes. La boucle externe sur les directions peut elle aussi donner lieu à une parallélisation.

Si $D$ est l'ensemble des directions considérées, et $D_1, D_2, ..., D_m$ des sous-ensembles formant une partition de $D$, alors on peut écrire

$$\begin{aligned} B(x) &= \sum_{\overrightarrow{\sigma} \in D} f(x, V(x, \overrightarrow{\sigma}))B(V(x, \overrightarrow{\sigma})) \\ &= \sum_{\overrightarrow{\sigma} \in D_1} f(x, V(x, \overrightarrow{\sigma}))B(V(x, \overrightarrow{\sigma})) \\ &\quad + \ldots \\ &\quad + \sum_{\overrightarrow{\sigma} \in D_m} f(x, V(x, \overrightarrow{\sigma}))B(V(x, \overrightarrow{\sigma})) \\ &= \sum_{i=1}^{m} \sum_{\overrightarrow{\sigma} \in D_i} f(x, V(x, \overrightarrow{\sigma}))B(V(x, \overrightarrow{\sigma})) \end{aligned}$$

Il est donc possible de créer un fil d'exécution par sous-ensemble de direction. À cette échelle, on peut cependant envisager d'utiliser plutôt une répartition sur grille de calcul. Chaque nœud de la grille prend alors en charge un sous-ensemble des directions. Cela implique évidemment un surcoût en mémoire et en communication :

- Chaque nœud doit posséder une copie de l'intégralité des voxels dans sa propre mémoire.
- après le traitement de ces sous-ensembles, les résultats doivent être fusionnées. La synchronisation s'effectue en ajoutant à chaque voxel la somme de tous les incréments de radiosité dûs à chaque nœud. Il doit

donc y avoir une communication globale $N \leftrightarrow N$ des valeurs de tous les voxels.

Pour un grand ensemble de directions, le coût des communications est largement compensé par le gain de performances (cf. section 6.2.3 page 145). Remarquons cependant que la synchronisation tardive en fin d'itération prive de l'utilisation de la relaxation de Gauss-Seidel habituellement préférée à Jacobi pour la rapidité de convergence (cf. section 2.3.3.3 page 56). En effectuant plusieurs synchronisations au cours d'une itération, on peut pallier légèrement cela.

Enfin, il est également possible de rajouter une parallélisation supplémentaire : chaque sous-ensemble de directions peut être subdivisé à son tour pour être localement traité par plusieurs threads.

### 6.2.2 Distribution des données

La radiosité est un processus global utilisant et modifiant l'intégralité de ses données. La répartition des voxels sur plusieurs machines, pour diviser la mémoire nécessaire au traitement d'une scène de grande taille, est donc une problématique difficile. De nombreuses communications doivent être envisagées pour donner à chaque nœud de calcul toutes les informations nécessaires à la visibilité et à la propagation. Des stratégies de répartition et de rééquilibrage ont donc été déployées pour une utilisation efficace d'une grille de calcul.

#### 6.2.2.1 Répartition des listes

La segmentation des voxels en groupes indépendants peut profiter de la présence des listes partitionnant l'espace. Une liste est caractérisée par deux entiers $(i, j)$, et pour une direction donnée, on extrait un tel couple $(i, j)$ de chaque voxel permettant de l'associer à une liste. Chaque machine peut donc se voir attribuer un sous-ensemble de listes (caractérisés par des intervalles de valeurs $[i_1; i_2]$ et $[j_1; j_2]$ pour $i$ et $j$), afin de recevoir les voxels correspondants. Les listes étant de tailles diverses, le nombre de listes attribuées à chaque machine peut varier pour répartir les voxels de façon à peu près équitable.

Il est donc possible, dans une direction donnée, de trouver une répartition équitable des voxels sur les machines. Cependant, un changement de direction implique une reconsidération totale des listes et une nouvelle distribution des voxels. La redistribution peut être réalisée par migration des voxels d'une machine à l'autre, et le problème principal revient à trouver une méthode permettant de minimiser les migrations.

Lors de la transition d'une direction à une direction proche, le $(i, j)$ de chaque voxel varie peu (sauf pour certains changements de direction modifiant le lien entre $i$, $j$ et deux des plans parmi $(Oxy)$, $(Oxz)$ et $(Oyz)$,

ayant lieu à chaque changement de la coordonnée « majeure » du vecteur directeur). Si $(i, j)$ varie peu, le voxel a de grandes chances de rester sur sa machine. Un ordre de parcours des directions de proche en proche permet donc de minimiser les migrations. Un tel ordre (Figure 6.6) remet cependant en cause le parcours « chaotique » choisi pour le raffinement progressif (cf. section 4.3.4 page 107).

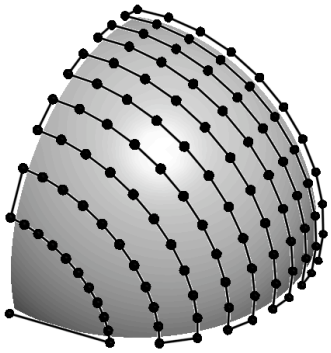


FIG. 6.6 – Parcours des directions de proche en proche sur un huitième de sphère.

#### 6.2.2.2 Rééquilibrage

Le rééquilibrage consistant à recalculer la répartition des listes parmi les machines peut être fait dynamiquement, plus ou moins régulièrement. La meilleure stratégie est toutefois difficile à trouver. D'un côté, le gain vient du fait qu'une répartition équitable évite à une machine d'être beaucoup plus lente que les autres et de retarder d'autant les barrières de synchronisations, ce qui correspondrait à du temps perdu sur les autres machines, mais d'un autre côté, le coût des communications ne doit pas faire perdre ce gain. La façon d'échanger les données, leur encapsulation et la régularité des rééquilibrages sont autant de paramètres étudiés par Rita Zrour pour évaluer les meilleurs compromis.

#### 6.2.2.3 Perspectives

La distribution des voxels peut encore être envisagée d'un point de vue géométrique, en découpant la scène en morceaux attribués aux différentes machines. Les listes calculées par chacun des nœuds sont alors des listes incomplètes, et des moyens de communication supplémentaires doivent être introduits pour que la phase de propagation puisse être effectuée efficacement. Ces travaux sont en cours d'investigation.

### 6.2.3 Résultats expérimentaux

La figure 6.7 expose le temps de calcul observé pour l'ensemble des répartitions/propagations lors de la résolution complète de la radiosité d'une scène. L'utilisation de plusieurs threads sur les machines bi-processeur et bi-cœur utilisées montre un gain sensible. Avec deux threads, le temps de calcul est diminué d'environ 25%. Ce gain s'amenuise lentement avec un nombre plus important de threads, du temps étant perdu par l'ordonnanceur du système.

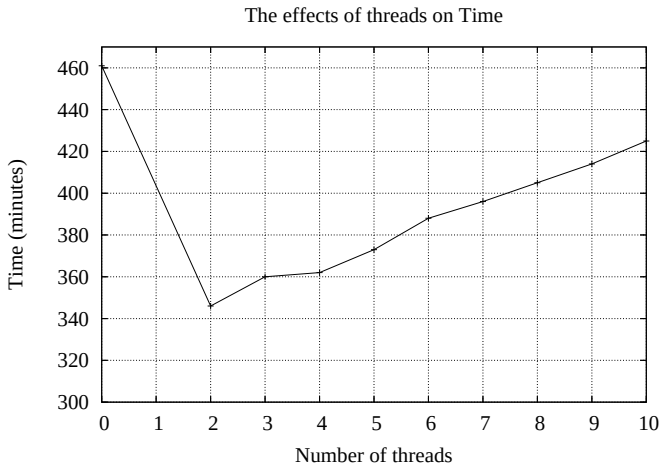



FIG. 6.7 – Efficacité des threads pour la répartition et la propagation. Sur les machines bi-processeur et bi-cœur utilisées, l'utilisation de deux threads est optimale.

La figure 6.8 page suivante expose le temps de calcul observé en fonction du nombre de machines se répartissant les directions. Le coût des synchronisations étant très faible comparativement au calcul de radiosité, la parallélisation divise pratiquement le temps de calcul par le nombre de machines, pour 2 et 3 nœuds. Il y a ensuite une saturation. Ces rapports dépendent du temps passé sur un sous-ensemble de directions. Plus ce temps est important (beaucoup de voxels), plus le rapport sera favorable.

La figure 6.9 page suivante expose le gain en temps de calcul dans le cas de la répartition des voxels. Le gain n'est pas très important à cause du coût des nombreuses communications nécessaires, mais il peut facilement diviser le temps de calcul par deux.

La figure 6.10 page 147 expose la répartition du temps passé en calcul effectif de radiosité et en travaux supplémentaires de communication et

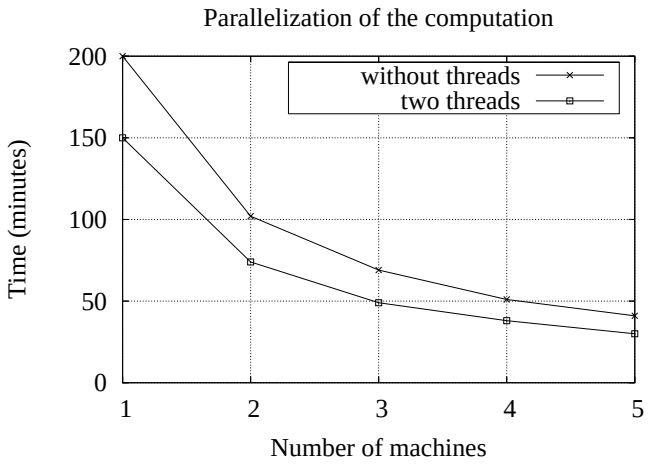



FIG. 6.8 – Pour la scène calculée, la répartition des directions divise pratiquement le temps de calcul par le nombre de machines disponibles si celui-ci est faible. Il y a ensuite saturation.

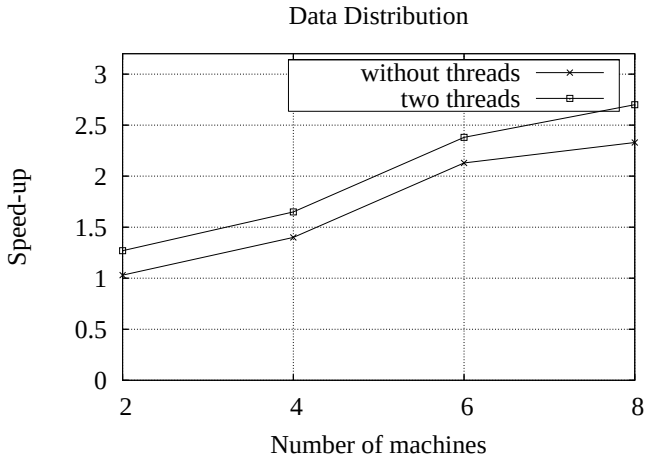



FIG. 6.9 – Les gains en temps de calcul observés pour la répartition des données sont faibles à cause du coût des nombreuses communications nécessaires, mais il peut facilement diviser le temps de calcul par deux.

rééquilibrage. On observe assez rapidement (4 machines) que les communications prennent environ 50% du temps de calcul.

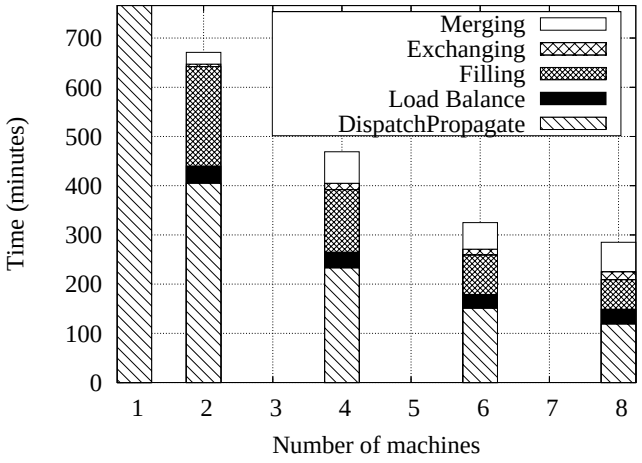



FIG. 6.10 – La parallélisation implique des phases de communication (échange et fusion de données) et de calculs annexes (équilibrage dynamique). La répartition du temps entre le calcul effectif et ces travaux supplémentaires dépend du nombre de machines.

## 6.3 Extensions du modèle d'illumination

La représentation par voxels utilisée par notre algorithme de radiosité est bien adaptée à quelques raffinements du calcul d'illumination, dans les cas de phénomènes eux-même volumiques. Le milieu participatif, interagissant avec les rayons, en est un exemple. De façon plus générale, il est également possible d'envisager une scène dont les objets ne sont pas des surfaces mais des volumes intangibles. Dans la section 4.5 page 114 traitant des limitations de notre algorithme de radiosité, nous avons souligné que du fait de sa nature volumique, cet algorithme présente des déficiences pour un travail efficace sur des données surfaciques. Un nuage de matière est un exemple de phénomène pour lequel une discrétisation en voxels semble plus adaptée. Nous nous sommes donc également penchés sur la problématique du transfert radiatif en météorologie.

### 6.3.1 Milieux participatifs

L'interaction la plus simple envisagée entre un rayon et le milieu traversé est une simple absorption, modélisable par une fonction de la distance parcourue dans le medium. La phase de propagation de notre algorithme passe de voxel en voxel pour transférer de l'énergie : on peut déduire des coordonnées de deux voxels voisins la distance parcourue par le rayon. On retrouve ici une caractéristique de la représentation par voxels d'une scène, qui est de modéliser implicitement les espaces vides, d'une façon similaire aux espaces pleins. Si $d$ est la distance parcourue dans un medium, on peut envisager d'utiliser une fonction de probabilité d'absorption de la forme $1 - e^{-\sigma d}$, $\sigma$ étant un paramètre dépendant de la densité du milieu.

Dans un souci de réalisme, il est envisageable de considérer une modélisation plus complexe de l'interaction des photons avec le milieu, sous forme de *in-scattering* et *out-scattering* (Figures 6.11 et 6.12 page ci-contre). Lors d'une absorption, plutôt que d'envisager une disparition du photon, on peut plutôt prévoir un éclatement, une dispersion alentours (*out-scattering*). À l'inverse, des rayons se concentrant en un seul point peuvent « ajouter » des photons à un rayon donné (*in-scattering*).

Autant l'absorption simple est facile à prendre en compte dans notre algorithme, autant *in* et *out-scattering* ne s'y insèrent pas facilement. Pour en tenir compte, il faudrait effectivement instancier des voxels dans les espaces vides, et leur faire enregistrer une émissivité dans différentes directions du fait de l'*out-scattering*. Une distribution de radiance (cf. chapitre 5 page 117) permet justement de répondre à ce problème. Le *out-scattering* étant ainsi représenté, le *in-scattering* devient implicite : il correspond à l'*out-scattering* des voisins.

Les équations régissant l'*out-scattering* dépendent fortement du milieu considéré. Nous nous sommes intéressés au cas des nuages météorologiques.

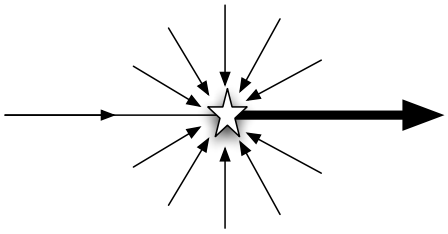


FIG. 6.11 – In-scattering : la lumière se concentre

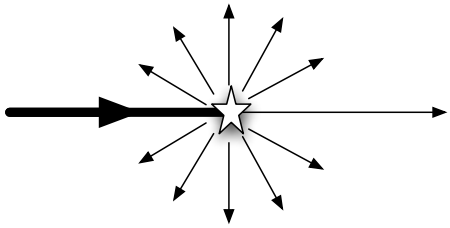


FIG. 6.12 – Out-scattering : la lumière se disperse

### 6.3.2 Application à la météorologie

Afin d'explorer le potentiel de notre algorithme dans un domaine inaccessible aux algorithmes de radiosité par patches, nous nous sommes penchés sur l'évaluation du transfert radiatif dans les nuages. Pour cela, nous n'avons pas effectué de recherches exhaustives sur le sujet, mais contacté un laboratoire de météorologie (en la personne de Frédéric Szczap du LaMP[1]) à même de nous fournir renseignements, données et jeux de tests.

#### 6.3.2.1 Transfert radiatif et fonctions de phase

Le transfert radiatif est un problème proche de l'illumination globale, en ce sens qu'il cherche à déterminer l'état d'un nuage sous un éclairement donné, en fonction des propriétés de diffusion interne du milieu. C'est un domaine de recherche important en météorologie [Len93]. Les lois de réémission du transfert radiatif sont modélisées par des *fonctions de phase*.

Une fonction de phase $\Phi$ définit pour un photon la probabilité de changer de direction. La probabilité pour un photon d'aller tout droit (0°) est forte, mais il est possible qu'il dévie. Dans certains cas, il est possible que la probabilité de faire demi-tour (180°) soit plus forte que de tourner de 90°. Deux fonctions de phases ont été portées à notre connaissance : celle de la théorie de Mie, et le modèle de Henyey-Greenstein. La fonction de phase de Mie nous a été fournie sous forme d'un jeu de coefficients assez complexe appliqué à des polynômes de Legendre. Elle est assez coûteuse à calculer. La fonction de Henyey-Greenstein est un modèle mathématique moins riche mais beaucoup plus simple, applicable à certains nuages modélisés de façon « idéale ». La fonction de phase de Henyey-Greenstein est la suivante, paramétrée par $g$ dont une valeur standard est $0,848$.

$$p_{HG}(\theta) = \frac{1}{2} \frac{1 - g^2}{(1 - 2 \times g \times \cos\theta + g^2)^{\frac{3}{2}}}$$

[1]Laboratoire de Météorologie Physique, Université Blaise Pascal, Clermont-Ferrand

On a

$$\frac{1}{2}\int_{\theta=0}^{\pi} p_{HG}(\theta) d\cos\theta = \frac{1}{2}\int_{\theta=0}^{\pi} p_{HG}(\theta)\sin\theta d\theta = 1$$

La figure 6.13 illustre la fonction de phase de Henyey-Greenstein dans un repère polaire.

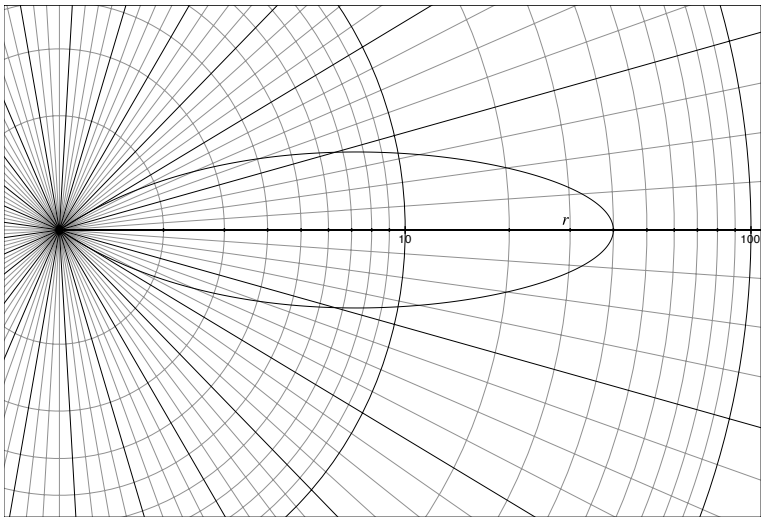



FIG. 6.13 – Fonction de phase de Henyey-Greenstein. C'est une distribution dont l'intégrale vaut 1.

#### 6.3.2.2 Expérimentations sur l'albedo

La modélisation du transfert radiatif permet notamment de calculer l'*albedo* d'un nuage. Il s'agit du rapport entre la lumière reçue et la lumière retransmise vers le haut.

Nous avons récupéré les résultats pour un modèle théorique utilisé par le laboratoire. La figure 6.14 page suivante montre l'albedo observé pour un éclairage zénithal sur un nuage dont on augmente l'épaisseur optique $\tau$. L'épaisseur optique $\tau$ du nuage est un paramètre affectant la probabilité d'interaction avec le milieu : en cas de *transmission directe* (non-interaction), le photon continue tout droit, sinon il est réémis dans une direction de probabilité définie par la fonction de phase. La probabilité de transmission directe est $e^{-\tau}$. $\tau$ est égal au produit entre l'*extinction* du milieu et la longueur de l'intersection entre le rayon lumineux et le morceau de milieu considéré.

L'extinction est considérée constante dans le nuage, et ce dernier est considéré infiniment long en bouclant sur les bords.

Nous avons modélisé un nuage sous forme de voxels d'extinction donnée, et appliqué une version modifiée de notre algorithme de radiosité pour effectuer un transfert radiatif utilisant les fonctions de phase. L'albedo que

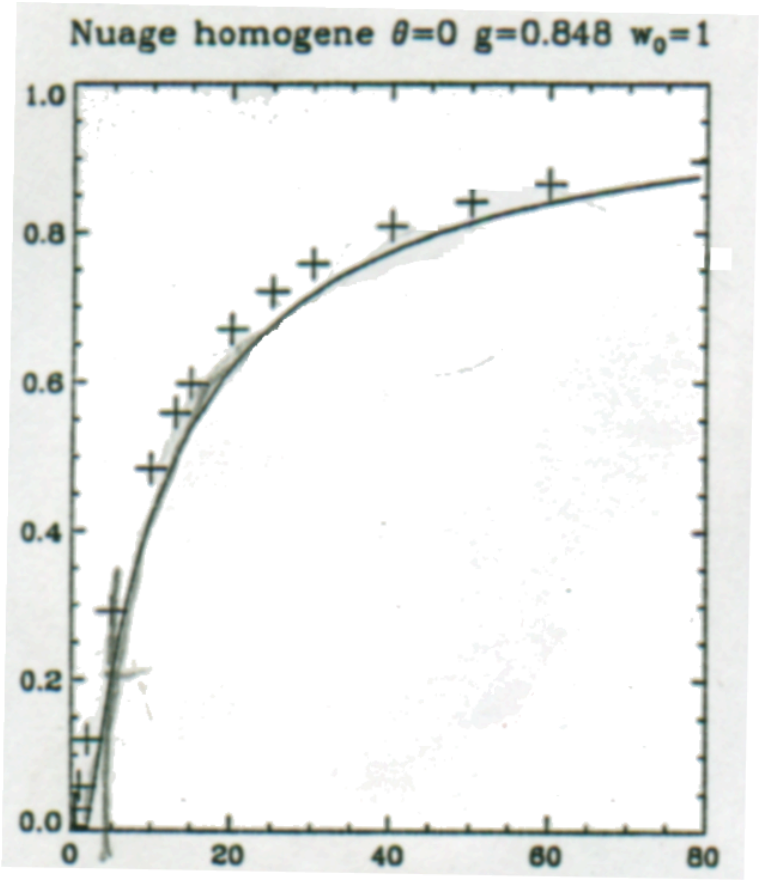



FIG. 6.14 – Albedo de référence pour un nuage homogène éclairé zénithalement, utilisant une fonction de phase de Henyey-Greenstein de paramètre g=0.848 ; l'albedo augmente rapidement avec l'épaisseur optique, puis se stabilise. Ces données nous ont été fournies par le LaMP. La courbe pleine provient d'un calcul par la méthode d'Eddington ; les croix sont les résultats d'une simulation de type Monte-Carlo.

nous avons observé est représenté figure 6.15 page suivante ; il est tout-à-fait semblable à l'albedo de référence de la figure 6.14.

#### 6.3.2.3 Perspectives

Pour utiliser notre algorithme de radiosité avec le transfert radiatif, du travail est encore à effectuer, même si ce premier résultat est encourageant. Il nous faut notamment développer une technique rapide pour simuler le fait que le nuage « boucle » sur les bords.

– Nous pensons utiliser des restes de division entière pour construire les listes représentant ce bouclage.

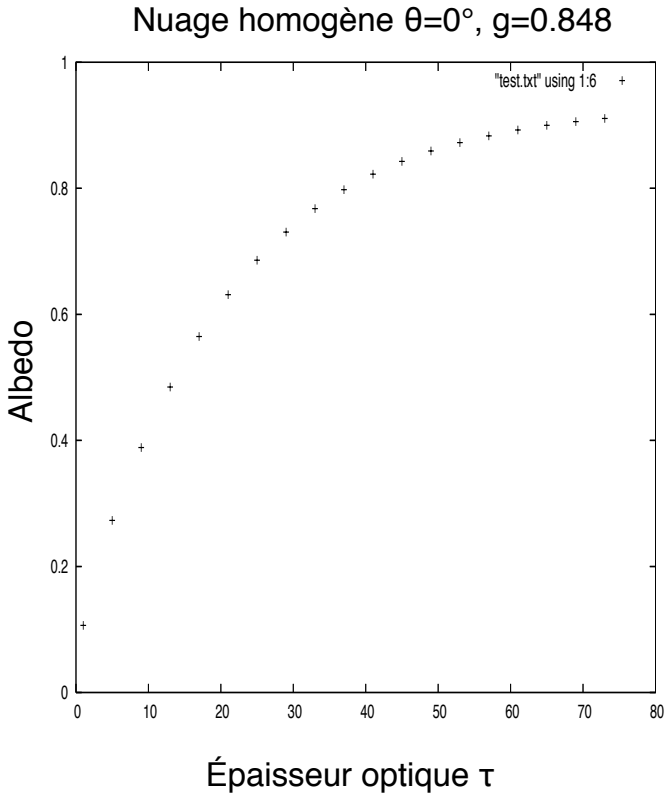



FIG. 6.15 – Albedo observé avec transfert radiatif calculé par notre algorithme de radiosité, à comparer avec la figure 6.14 page précédente

- Il nous faut également calculer précisément la longueur de l'intersection entre un rayon et un voxel quelconque pour calculer la probabilité de transmission directe de façon fiable, notamment pour un éclairage d'incidence non zénithale.
- Notons enfin que dans le cas des nuages, aucune notion de normale en chaque voxel n'est envisagée. De ce fait, les distributions de radiance, que nous utilisons pour représenter l'émission de chaque voxel dans toutes les directions, sont toujours considérées dans un repère global ne nécessitant pas de rotation. Il est donc envisageable d'utiliser les harmoniques sphériques plutôt que notre technique d'interpolation pour représenter les fonctions de phase au mieux.
- Les simulations effectuées par le LaMP sont réalisées grâce au logiciel SHDOM, qui utilise les harmoniques sphériques. Une étude du code de SHDOM est nécessaire pour justifier les gains de notre méthode.

Au final, nous avons montré que notre algorithme de radiosité pouvait sans doute être appliqué au transfert radiatif, mais il faut un nombre plus important de modifications que nous ne l'aurions cru.

## 6.4 Projet logiciel

### 6.4.1 Modularité

Notre algorithme de radiosité a été développé comme un module de calcul portable, aisément transposable sur une grille de calcul pour la problématique de la parallélisation.

Pour effectuer les visualisations, il a d'abord été développé pour communiquer avec le logiciel French3D également développé au laboratoire. De façon plus générale, de nombreux modules sont en développement et représentent un ensemble de briques logicielles restant à assembler (Figure 6.16).

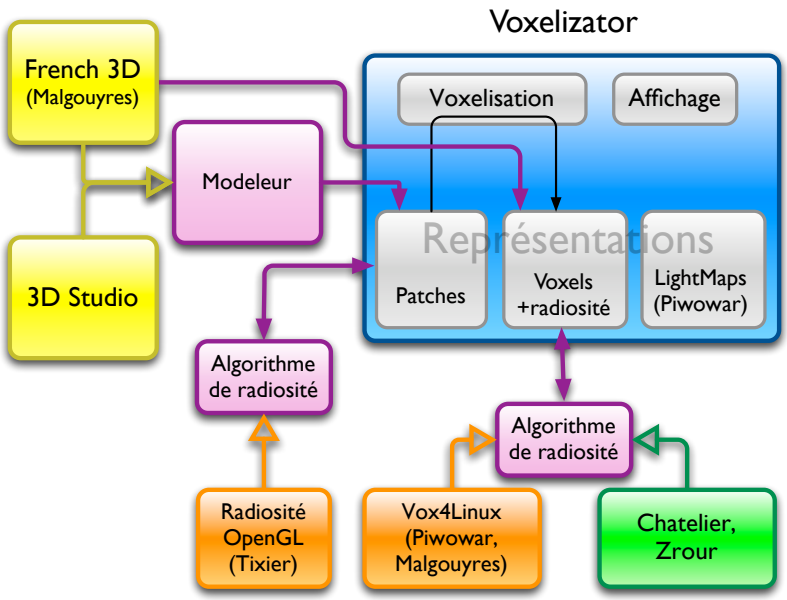



FIG. 6.16 – Projet logiciel

- différents algorithmes de radiosité sont développés : la radiosité par voxels, une implémentation utilisant OpenGL pour le calcul des facteurs de forme, et une technique utilisant le lancer de rayon (Vox4Linux).
- L'algorithme utilisant OpenGL utilise une représentation en patches de la scène, tandis que les deux autres utilisent une représentation par voxels.
- Le logiciel *Voxelizator* développé par Lukasz Piwowar permet de lire des scènes modélisées par patches, de les discrétiser et d'effectuer une visualisation efficace.
- French3D sert alors de modeleur pour fournir au *Voxelizator* des scènes à discrétiser. *Voxelizator* peut également lire des scène modélisées sous 3D Studio.

### 6.4.2 Le Voxelizator

Le programme *Voxelizator* est devenu central dans le projet logiciel. Des méthodes mixtes d'affichage, utilisant à la fois résultats de radiosité et de lancer de rayons, sont en cours d'élaboration. Cela permettra à terme de produire des images de bonne qualité tout en conservant un nombre de voxels et de directions assez faibles. Le principe de l'*Instant Radiosity* (cf. section 2.2.7 page 48) doit notamment être repris pour effectuer des passes de propagation de radiosité dans le modèle discret de la scène, afin de produire l'éclairage indirect faisant défaut au ray-tracing simple.

La figure 6.17 page ci-contre est un exemple de ce que l'on peut obtenir actuellement. L'obtention de cette image de la scène *soda.mgf*[1] nécessite actuellement plusieurs étapes :

- discrétisation de la scène ;
- export de la scène discrète sur la machine de calcul ;
- calcul de la radiosité ;
- import de la solution dans le *Voxelizator* ;
- ray-tracing utilisant les données de radiosité.

À terme, le logiciel permettra de réaliser ces étapes plus simplement et plus rapidement. Il devrait aussi nous permettre de mieux connaître les valeurs nécessaires et suffisantes des paramètres de l'algorithme pour obtenir des résultats corrects à moindre coût. En répartissant la génération de l'image entre ray-tracing et radiosité, les défauts respectifs doivent se compenser et ne pas générer d'artifacts gênants.

[1] scène classique disponible à l'URL `http://radsite.lbl.gov/mgf/scene/soda.mgf.Z`

FIG. 6.17 – Les données de radiosité sont intégrées au ray-tracer du Voxelizator

## 6.5 Conclusion

L'algorithme de radiosité que nous avons présenté au chapitre 4 se prête bien à de nombreuses améliorations. Utilisé de façon duale avec une représentation géométrique en patches, il permet de mixer les deux approches et d'être utilisé comme outil privilégié de calcul des visibilités.

De plus, dans le cas d'un jeu de données très lourd, ces calculs peuvent assez facilement être répartis sur plusieurs machines en une parallélisation efficace. La parallélisation peut avoir lieu à plusieurs niveaux et les gains sont intéressants.

L'utilisation de l'algorithme dans le domaine du transfert radiatif en météorologie montre que l'on peut étendre son champ d'action hors de la seule synthèse d'image. Les nuages constituent un exemple idéal de scène pour laquelle les voxels sont une représentation compacte et efficace.

L'algorithme est maintenant prêt à être implanté dans un logiciel permettant d'effectuer des comparaisons pertinentes avec les méthodes de radiosité plus classiques. Cela doit aussi permettre d'optimiser l'utilisation des paramètres pour obtenir des résultats satisfaisants à moindre coût.

# Conclusion et Perspectives

La grande diversité des phénomènes physiques à prendre en compte dans la quête du réalisme en synthèse d'image mène à une profusion d'algorithmes. Dans le seul domaine de l'illumination globale, en différenciant les modèles par facteurs de forme, hémicube, collocation, Monte-Carlo Light-tracing, Bidirectional Path Tracing, Random Walk, Photons Maps, Instant Radiosity, Metropololis... on dénombre facilement une dizaine d'algorithmes. Si leurs modes de propagation de l'énergie sont variés, leurs façons d'appréhender la visibilité se limitent toutefois à deux approches : la projection ou le lancer de rayon. Il n'y a pas eu de réelle nouveauté lors de l'introduction des variations algorithmiques. Les développements ont surtout eu lieu en raffinant les méthodes d'exploration. On peut par exemple voir le modèle Metropolis comme une perturbation du Bidirectional Path-tracing, ou l'Instant Radiosity comme une utilisation poussée des photon-maps.

Le principal apport de cette thèse est la création d'un algorithme de radiosité inédit. La propagation de l'énergie s'effectue bien sur des rayons échantillonnant les directions de l'espace, mais le mode de construction de ces rayons est très différent du lancer de rayon classique, qu'il soit continu ou discret. En combinant voxels et arithmétique en géométrie discrète, nous avons pu inventer une méthode de calcul de complexité quasi linéaire, donc quasi optimale vu le problème posé. L'originalité repose sur le fait que l'information de visibilité n'est pas directement cherchée, mais déduite d'un agencement purement mathématique des données.

La géométrie discrète est un domaine bien plus vaste que la seule construction de droites 3D, même si nous n'avons utilisé pratiquement que cette notion. Si la discrétisation garantit que les surfaces sont des surfaces de Jordan, les droites 3D utilisées comme rayons sont topologiquement cohérentes, ce qui donne un sens aux « intersections » constatées. L'arithmétique est également sans doute capable d'apporter une solution élégante aux problèmes nous restant à régler pour le transfert radiatif.

Le deuxième apport important de cette thèse est une représentation simple et efficace de fonctions 2D sphériques lisses, utilisables pour encoder les distributions de radiance des BRDFs. Combinée au principe de calcul de visibilité déployé pour la radiosité, cette représentation permet de remonter à un algorithme d'illumination globale. La représentation choisie, à base de points de contrôles, est plus simple et plus rapide que les techniques habituelles, et le surcoût induit par son utilisation au lieu de la seule radiosité est assez raisonnable. Il a fallu pour cela trouver une façon efficace d'appréhender chaque opération à appliquer sur la fonction sphérique représentée.

Les limitations de notre algorithme se trouvent dans le domaine d'application choisi. Si la scène n'est pas adaptée à une représentation en voxels, c'est-à-dire qu'elle dispose de grandes surfaces planes et fines nécessitant de petits – et donc de nombreux – voxels, alors l'algorithme est mis en défaut.

Les scènes denses lui donnent en revanche un intérêt accru. Grâce à sa faible complexité et à des temps de calculs stables et prévisibles, l'algorithme est cependant bien adapté à un calcul volontairement lourd, par exemple pour améliorer les résultats numériques plus que visuels.

Les perspectives d'évolution de l'algorithme sont assez importantes. La piste la plus intéressante à suivre dans un premier temps est la parallélisation qui peut être injectée à de nombreux niveaux. Les capacités du cœur de calcul à être parallélisé permettent de répartir assez aisément le travail sur plusieurs machines et obtenir des gains appréciables. La parallélisation peut avoir lieu sur les données ou sur les tâches, et autorise à utiliser différentes techniques simultanément : par threads et sur plusieurs machines.

La représentation des fonctions sphériques nous a également été utile pour l'utilisation des fonctions de phase dans le transfert radiatif, application dérivée de notre algorithme, que nous avons abordée pour le tester sur des données très denses. Ce travail mériterait toutefois d'être approfondi, car il nécessite une étude plus poussée des mécanismes du programme SHDOM utilisé habituellement en météorologie, avec lesquelles il faudrait comparer les résultats.

Enfin, même si les expérimentations n'ont pas encore été menées pour comparer les productions de notre algorithme avec les logiciels classiques d'imagerie de synthèse, le logiciel Voxelizator doit nous permettre de les effectuer dans un proche avenir. En disposant d'un environnement convivial, capable d'appliquer des méthodes mixtes de visualisation, nous pourrons affiner notre utilisation des paramètres pour obtenir des images de qualité en un temps raisonnable.

# Bibliographie


[AAS97] E. Andres, R. Acharya, and C. Sibata. Discrete analytical hyperplanes. *Graphical Models and Image Processing*, 59(5) :302–309, Septembre 1997.

[ABM01] D. Arques, V. Biri, and S. Michelin. A New Mathematical Development for Radiosity Animation with Galerkin. In *Computer Animation 2001*, pages 217–255, November 2001.

[ACS00] K. Atkinson, Da-Kwon Chien, and Jaehoon Seol. Numerical analysis of the radiosity equation using the collocation method. *Electronic Transactions on Numerical Analysis*, 11, 2000.

[ADL05] Eric Andres, Guillaume Damiand, and Pascal Lienhardt, editors. *Discrete Geometry for Computer Imagery, 12th International Conference, DGCI 2005, Poitiers, France, April 13-15, 2005, Proceedings*, volume 3429 of *Lecture Notes in Computer Science*. Springer, 2005.

[AJ97] Eric Andres and Marie-Andrée Jacob. The discrete analytical hyperspheres. *IEEE Transactions on Visualization and Computer Graphics*, 3(1) :75–86, 1997.

[AM96] D. Arquès and S. Michelin. Proximity radiosity : exploiting coherence to accelerate form factor computations. In *Proceedings of the eurographics workshop on Rendering techniques '96*, pages 143–ff., London, UK, 1996. Springer-Verlag.

[And00] E. Andres. *Modélisation Analytique Discrète d'Objets Géométriques.* Habilitation à diriger des recherches, UFR Sciences Fondamentale et Appliquées - Université de Poitiers (France), Décembre 2000.

[And03] E. Andres. Discrete linear objects in dimension n : The standard model. *Graphical Models*, 65 :92–111, 2003.

[AS00] Michael Ashikhmin and Peter Shirley. An anisotropic Phong BRDF model. *Journal of Graphics Tools : JGT*, 5(2) :25–32, 2000.

[ATCF90] III A. T. Campbell and Donald S. Fussell. Adaptive mesh generation for global diffuse illumination. *SIGGRAPH Comput. Graph.*, 24(4) :155–164, 1990.

[AW87] John Amanatides and Andrew Woo. A fast voxel traversal algorithm for ray tracing. In *Eurographics '87*, pages 3–10. Elsevier Science Publishers, Amsterdam, North-Holland, 1987.

[BB04] Valentin E. Brimkov and Reneta P. Barneva. Connectivity of discrete planes. *Theor. Comput. Sci.*, 319(1-3) :203–227, 2004.

[BE92] Bern and Eppstein. Mesh generation and optimal triangulation. In *Computing in Euclidean Geometry, Edited by Ding-Zhu Du and Frank Hwang, World Scientific, Lecture Notes Series on Computing – Vol. 1.* World Scientific Publishing Company, 1992.

[BFB97] Miguel A. Blanco, M. Florez, and M. Bermejo. Evaluation of the rotation matrices in the basis of real spherical harmonics. *J. Molecular Structure (Theochem)*, 419 :19–27, 1997.

[BIS98] A. Bar-Lev, A. Itzkovitz, and A. Raviv A. Schuster. Parallel vertex-to-vertex radiosity on a distributed shared memory. In *Proc. of the Fifth Int'l Symposium on Solving Irregularly Structured Problems in Parallel (IRREGULAR '98)*, 1998.

[BM02] Jasmine Burguet and Rémy Malgouyres. Multi-scale discrete surfaces. In *DGCI '02 : Proceedings of the 10th International Conference on Discrete Geometry for Computer Imagery*, pages 338–349, London, UK, 2002. Springer-Verlag.

[BMA03] Venceslas Biri, Sylvain Michelin, and Didier Arquès. Dynamic radiosity using higher order functions bases and temporal coherence. In *WSCG*, 2003.

[BMP93] K. Bouatouch, D. Menard, and T. Priol. Parallel radiosity using a shared virtual memory. In *Proceedings of ATARV'93*, pages 71–83, 1993.

[Buz03] Lilian Buzer. A linear incremental algorithm for naive and standard digital lines and planes recognition. *Graph. Models*, 65(1-3) :61–76, 2003.

[BW97] Jules Bloomenthal and Brian Wyvill, editors. *Introduction to Implicit Surfaces.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[CB98] M. Couprie and G. Bertrand. Simplicity surfaces : a new definition of surfaces in z3. *SPIE Vision Geometry VII*, 3454 :40–51, 1998.

[CG85] Michael F. Cohen and Donald P. Greenberg. The hemi-cube : a radiosity solution for complex environments. In *SIGGRAPH '85 : Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 31–40. ACM Press, 1985.

[cga] Cgal, Computational Geometry Algorithms Library. http ://www.cgal.org.

[Che87] L. P. Chew. Constrained delaunay triangulations. In *SCG '87 : Proceedings of the third annual symposium on Computational geometry*, pages 215–222, New York, NY, USA, 1987. ACM Press.

[CM05] Pierre Y. Chatelier and Rémy Malgouyres. A low complexity discrete radiosity method. In Andres et al. [ADL05], pages 392–403.

[CM06] Pierre Y. Chatelier and Remy Malgouyres. A low-complexity discrete radiosity method. *Computers & Graphics*, 30 :37–45, February 2006.

[COK97] Daniel Cohen-Or and Arie Kaufman. 3D line voxelization and connectivity control. *IEEE Computer Graphics and Applications*, 17(6) :80–87, /1997.

[CPB06] Luc Claustres, Mathias Paulin, and Yannick Boucher. A wavelet-based framework for acquired radiometric quantity representation and accurate physical rendering. *The Visual Computer*, pages 221–237, 2006.

[cXSAWG91] François X. Sillion, James R. Arvo, Stephen H. Westin, and Donald P. Greenberg. A global illumination solution for general reflectance distributions. In *SIGGRAPH '91 : Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 187–196, New York, NY, USA, 1991. ACM Press.

[DBB02] Philip Dutré, Kavita Bala, and Philippe Bekaert. *Advanced Global Illumination*. A. K. Peters, Ltd., Natick, MA, USA, 2002.

[DBB03] Philip Dutré, Philippe Bekaert, and Kavita Bala. *Advanced Global Illumination*. A K Peters, Natick, USA, 2003.

[DR95] Isabelle Debled-Rennesson. *Étude et reconnaissance des droites et plans discrets*. PhD thesis, Université Louis Pasteur, Strasbourg, 1995.

[ED06] Elmar Eisemann and Xavier Décoret. Fast scene voxelization and applications. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 71–78. ACM SIGGRAPH, 2006.

[FC94] François X. Sillion and Claude Puech. *Radiosity & Global Illumination*. Morgan Kaufmann Publishers, Inc., 1994.

[FG99] Pascal Jean Frey and Paul-Louis George. *Maillages : Applications aux éléments finis*. Hermes, 1999.

[Fun96] Thomas A. Funkhouser. Coarse-grained parallelism for hierarchical radiosity using group iterative methods. *Computer Graphics*, 30(Annual Conference Series) :343–352, 1996.

[GA93] Irene Gargantini and H. H. Atkinson. Ray tracing an octree : Numerical evaluation of the first interaction. *Comput. Graph. Forum*, 12(4) :199–210, 1993.

[Gla94] Andrew S. Glassner. "a model for fluorescence and phosphorescence". In Stefan Haas, Stefan Muller, Georg Sakas, and Peter Shirley Proceedings, editors, *Fifth Eurographics Workshop on Rendering*, pages 57–68. Springer-Verlag, June 1994.

[GRS95] Pascal Guitton, Jean Roman, and Gilles Subrenat. A parallel method for progressive radiosity. Technical Report Research Report 992-95, LaBRI, Talence, France, 1995.

[Guo95] Baining Guo. A multiscale model for structure-based volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(4) :291–301, 1995.

[GWS04] Johannes Günther, Ingo Wald, and Philipp Slusallek. Realtime Caustics using Distributed Photon Mapping. In *Rendering Techniques 2004, Proceedings of the Eurographics Symposium on Rendering*, pages 111–121, June 2004.

[Han] Olaf Hansen. On the stability of the collocation method for the radiosity equation on polyhedral domains. citeseer.ifi.unizh.ch/hansen00stability.html.

[Han03] Olaf Hansen. On the properties of the radiosity equation near corners. *PAMM*, 2(1) :414–415, 2003.

[Hec93] Paul S. Heckbert. Finite Element Methods for Radiosity. In *ACM SIGGRAPH '93 Course Notes - Global Illumination*, pages 1–7. 1993.

[Her92] Gabor T. Herman. Discrete multidimensional jordan surfaces. *CVGIP : Graph. Models Image Process.*, 54(6) :507–515, 1992.

[HK93] Pat Hanrahan and Wolfgang Krueger. Reflection from layered surfaces due to subsurface scattering. In *SIGGRAPH '93 : Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 165–174, New York, NY, USA, 1993. ACM Press.

[How82] John R. Howell. *A Catalog of Radiation Configuration Factors*. MH, NY, 1982.

[HSA91] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. In *SIGGRAPH '91 : Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 197–206, New York, NY, USA, 1991. ACM Press.

[IK00] Frank Dachille IX and Arie Kaufman. Incremental triangle voxelization. In *Graphics Interface*, pages 205–212, May 2000.

[JC98] Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *SIGGRAPH '98 : Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 311–320, New York, NY, USA, 1998. ACM Press.

[Jen96] Henrik Wann Jensen. Global Illumination Using Photon Maps. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 21–30, New York, NY, 1996. Springer-Verlag/Wien.

[JJS93] N. S. Jayant, J. Johnston, and R. Safranek. Signal compression based on models of human perception. *Proc. IEEE*, 81(10), october 1993.

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

[KBT03] Marcelo Kallmann, Hanspeter Bieri, and Daniel Thalmann. Fully dynamic constrained delaunay triangulations. In G. Brunnett, B. Hamann, H. Mueller, and L. Linsen, editors, *Geometric Modelling for Scientific Visualization*, pages 241–257. Springer-Verlag, Heidelberg, Germany, first edition, 2003. ISBN 3-540-40116-4.

[Kel97] Alexander Keller. Instant radiosity. In *SIGGRAPH '97 : Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[KKB+05] Jaroslav Křivánek, Jaakko Konttinen, Kadi Bouatouch, Sumanta Pattanaik, and Jiří Žára. Fast approximation to spherical harmonic rotation. In *SCCG '06 : Proceedings of the 22nd spring conference on Computer graphics (to appear)*, New York, NY, USA, 2005. ACM Press.

[KM99] Jan Kautz and Michael D. McCool. Interactive rendering with arbitrary brdfs using separable approximations. In *SIG-*

*GRAPH '99 : ACM SIGGRAPH 99 Conference abstracts and applications*, page 253, New York, NY, USA, 1999. ACM Press.

[KR] Peter Kostele and Dan Rockmore. The SpharmonicKit and S2Kit libraries. http ://www.cs.dartmouth.edu/ geelong/sphere.

[KS87] Arie Kaufman and Eyal Shimony. 3d scan-conversion algorithms for voxel-based graphics. In *SI3D '86 : Proceedings of the 1986 workshop on Interactive 3D graphics*, pages 45–75, New York, NY, USA, 1987. ACM Press.

[KSS02] Jan Kautz, Peter-Pike Sloan, and John Snyder. Fast, arbitrary brdf shading for low-frequency lighting using spherical harmonics. In *EGRW '02 : Proceedings of the 13th Eurographics workshop on Rendering*, pages 291–296, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[KvDS96] Jan J. Koenderink, Andrea J. van Doorn, and Marigo Stavridi. Bidirectional reflection distribution function expressed in terms of surface scattering modes. In *ECCV '96 : Proceedings of the 4th European Conference on Computer Vision-Volume II*, pages 28–39, London, UK, 1996. Springer-Verlag.

[Len93] Jacqueline Lenoble. *Atmospheric Radiative Transfer*, volume 127. A. Deepak Publishing, 1993.

[LRP97] G. W. Larson, H. Rushmeier, and C. Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Transactions on Visualization and Computer Graphics*, 3(4) :291–306, /1997.

[LSW99] Shunlin Liang, Alan H. Strahler, and Charles Walthall. Retrieval of land surface albedo from satellite observations : A simulation study. *Journal of Applied Meteorology*, june 1999.

[LW94] Eric P. Lafortune and Yves D. Willems. A theoretical framework for physically based rendering. *Comput. Graph. Forum*, 13(2) :97–107, 1994.

[LW99] C. Lincke and C. A. Wüthrich. An exact weaving rasterization algorithm. In V. Skala, editor, *WSCG'99 Conference Proceedings*, 1999.

[LW00] Christoph Lincke and Charles A. Wüthrich. Towards a unified approach between digitizations of linear objects and discrete analytical objects. In *WSCG*, 2000.

[Mal96] R. Malgouyres. There is no local characterization of separating and thin objects in $\mathbb{Z}^3$. *Theoretical Computer Science*, 163 :303–308, 1996.

[Mal97] R. Malgouyres. A definition of surfaces of $\mathbb{Z}^3|$ a new 3d discrete jordan theorem. *Theoretical Computer Science*, 186 :1–41, 1997.

[Mal02] Rémy Malgouyres. A discrete radiosity method. In Achille Braquelaire, Jacques-Olivier Lachaud, and Anne Vialard, editors, *Discrete Geometry for Computer Imagery, 10th International Conference, DGCI 2002, Bordeaux, France*, pages 428–438. Springer, April 2002.

[Mal05] Rémy Malgouyres. *Algorithmes pour la synthèse d'images et l'animation 3D, $2^{ème}$ édition*. Dunod, 2005.

[MB99] R. Malgouyres and G. Bertrand. A new local property of strong $n$-surfaces. *Pattern Recognition Letters*, 20 :417–428, 1999.

[MGM06] T. Marzais, Y. Gérard, and R. Malgouyres. $lp-$fitting approach for reconstructing parametric surfaces from point clouds. In *International Conference on Computer Graphics Theory and Applications, Setúbal, Portugal*, pages 325–330, February 2006.

[MKW+04] Gerd Marmitt, Andreas Kleer, Ingo Wald, Heiko Friedrich, and Philipp Slusallek. Fast and Accurate Ray-Voxel Intersection Techniques for Iso-Surface Ray Tracing. In *Vision, Modelling, and Visualization 2004 (VMV) , November 16-18, Stanford (CA), USA*, 2004.

[MR81] David G. Morgenthaler and Azriel Rosenfeld. Surfaces in three-dimensional digital images. *Information and Control*, 51(3) :227–247, 1981.

[Neu01] Attila Neumann. *Constructions of Bidirectional Reflection Distribution Functions*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, 2001.

[Öhr] Manne Öhrström. Spherical harmonics, precomputed radiance transfer and realtime radiosity in computer games. http ://citeseer.ist.psu.edu/685526.html.

[PC94] Derek J. Paddon and Alan Chalmers. Parallel processing of the radiosity method. *Computer-Aided Design*, 26(12) :917–927, 1994.

[Per94] L. Perroton. *Segmentation Parallèle d'Images Volumiques*. PhD thesis, Ecole Normale Supérieure de Lyon, December 1994.

[PM95] S. N. Pattanaik and S. P. Mudur. Adjoint equations and random walks for illumination computation. *ACM Trans. Graph.*, 14(1) :77–102, 1995.

[Poy98] Charles Poynton. The rehabilitation of gamma. In Bernice E. Rogowitz and Thrasyvoulos N. Pappas, editors, *SPIE Conf. Human Vision and Electronic Imaging III*, volume 3299, pages 232–249, P.O. Box 10, Bellingham, Washington, 98227-0010, USA, January 1998. SPIE–The International Society for Optical Engineering.

[PPS97] Frederic Perez, Xavier Pueyo, and Francois X. Sillion. Global illumination techniques for the simulation of participating media. In Julie Dorsey and Phillipp Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 309–320, New York, NY, 1997. Springer Wien.

[PRR98] Axel Podehl, Thomas Rauber, and Gudula Rünger. A shared-memory implementation of the hierarchical radiosity method. *Theoretical Computer Science*, 196(1–2) :215–240, 1998.

[PTYG00] Sumanta N. Pattanaik, Jack Tumblin, Hector Yee, and Donald P. Greenberg. Time-dependent visual adaptation for fast realistic image display. In *SIGGRAPH '00 : Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 47–54, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[Raf] Christophe Raffali. GlSurf mathes et dessins en objective-caml. http ://www.lama.univ-savoie.fr/ftp/pub/users/RAFFALLI/Papers/GlSurf.pdf.

[RAPP97] Luc Renambot, Bruno Arnaldi, Thierry Priol, and Xavier Pueyo. Towards efficient parallel radiosity for dsm-based parallel computers using virtual interfaces. In *PRS '97 : Proceedings of the IEEE symposium on Parallel rendering*, pages 79–86, New York, NY, USA, 1997. ACM Press.

[Rev91] Jean-Pierre Reveillès. *Géometrie discrète, calcul en nombres entiers et algorithmique*. PhD thesis, Université Louis Pasteur, Strasbourg, 1991.

[Rus97] Szymon Rusinkiewicz. A survey of brdf representation for computer graphics, 1997. http ://www.cs.princeton.edu/ smr/cs348c-97/surveypaper.html.

[Rus98] Szymon Rusinkiewicz. A new change of variables for efficient BRDF representation. In G. Drettakis and N. Max, editors, *Rendering Techniques '98 (Proceedings of Eurographics Rendering Workshop '98)*, pages 11–22, New York, NY, 1998. Springer Wien.

[SAG94] Brian Smits, James Arvo, and Donald Greenberg. A clustering algorithm for radiosity in complex environments. In *SIGGRAPH '94 : Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 435–442, New York, NY, USA, 1994. ACM Press.

[SAS92] Brian E. Smits, James R. Arvo, and David H. Salesin. An importance-driven radiosity algorithm. *SIGGRAPH Comput. Graph.*, 26(2) :273–282, 1992.

[Sbe93] Mateu Sbert. An integral geometry based method for fast form-factor computation. *Computer Graphics Forum (Eurographics '93)*, 12(3) :409–420, 1993.

[SC95] Nilo Stolte and René Caubet. Discrete ray-tracing of huge voxel spaces. *Comput. Graph. Forum*, 14(3) :383–394, 1995.

[Seo02] Jaehoon Seol. *Analysis of the radiosity equation using the collocation method*. Phd, Graduate College of The University of Iowa, dec 2002.

[SGS+02] Crystal B. Schaaf, Feng Gao, Alan H. Strahler, Wolfgang Lucht ang Xiaowen Li, Trevor Tsang, Nicholas C. Strugnell, Xiaoyang Zhang, Yufang Jin, and Jan-Peter Muller. First operational brdf, albedo and nadir reflectance products from modis. *Remote Sensing of Environment*, 83 :135–148, november 2002.

[SH00] François Sillion and Jean-Marc Hasenfratz. Efficient parallel refinement for hierarchical radiosity on a dsm computer. In *Third Eurographics Workshop on Parallel Graphics and Visualisation, Girona*, pages 61–74, Sep 2000.

[Shi96] Peter Shirley. Monte Carlo Methods for Rendering. In *ACM SIGGRAPH '96 Course Notes CD-ROM - Global Illumination in Architecture and Entertainment*, pages 1–26. 1996.

[Siv04] Isabelle Sivignon. *De la caractérisation des primitives à la reconstruction polyédrique de surfaces en géométrie discrète.* PhD thesis, Laboratoire des Images et des Signaux, INPG, Grenoble, 2004.

[SK98] M. Sramek and A. Kaufman. Object voxelization by filtering. In *IEEE Symposium on Volume Visualization*, pages 111–118, 1998.

[SK99] M. Sramek and A. Kaufman. Alias-free voxelization of geometric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3) :251–266, 1999.

[SK00] M. Sramek and A. Kaufman. Vxt : a c++ class library for object voxelization. *Volume Graphics*, pages 119–134, 2000.

[SK01] Nilo Stolte and Arie Kaufman. Novel techniques for robust voxelization and visualization of implicit surfaces. *Graph. Models*, 63(6) :387–412, 2001.

[SS95] Peter Schröder and Wim Sweldens. Spherical wavelets : efficiently representing functions on the sphere. *Computer Graphics*, 29(Annual Conference Series) :161–172, 1995.

[Sta99] Jos Stam. Diffraction shaders. In Alyn Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 101–110, Los Angeles, 1999. Addison Wesley Longman.

[SW94] Wolfgang Stürzlinger and Christoph Wild. Parallel visibility computations for parallel radiosity. In *CONPAR 94 - VAPP VI : Proceedings of the Third Joint International Conference on Vector and Parallel Processing*, pages 405–413, London, UK, 1994. Springer-Verlag.

[TR93] Jack Tumblin and Holly Rushmeier. Tone reproduction for realistic images. *IEEE Comput. Graph. Appl.*, 13(6) :42–48, 1993.

[USH82] J.K. Udupa, S.N. Srihari, and G.T. Herman. Boundary detection in multidimensions. *PAMI*, 4(1) :41–50, January 1982.

[Vea94] Eric Veach. Bidirectional estimators for light transport. In *Fifth Eurographics Workshop on Rendering*, pages 147–162, June 1994.

[VG97] Eric Veach and Leonidas J. Guibas. Metropolis light transport. *Computer Graphics*, 31(Annual Conference Series) :65–76, 1997.

[Vit99] J. Vittone. *Caractérisation et reconnaissance de droites et de plans en géométrie*. PhD thesis, Université Joseph Fourier, Grenoble,France, 1999.

[War92] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *SIGGRAPH '92 : Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 265–272, New York, NY, USA, 1992. ACM Press.

[War94] Greg Ward. A contrast-based scalefactor for luminance display. pages 415–421, 1994.

[Wes94] Ruediger Westermann. A multiresolution framework for volume rendering. In Arie Kaufman and Wolfgang Krueger, editors, *1994 Symposium on Volume Visualization*, pages 51–58, 1994.

[WH92] Gregory J. Ward and Paul Heckbert. Irradiance Gradients. In *Third Eurographics Workshop on Rendering*, pages 85–98, Bristol, UK, 1992.

[WPS+03] Ingo Wald, Timothy J. Purcell, Joerg Schmittler, Carsten Benthin, and Philipp Slusallek. Realtime Ray Tracing and its use for Interactive Global Illumination. In *Eurographics State of the Art Reports*, 2003.

[WRC88] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88 : Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 85–92, New York, NY, USA, 1988. ACM Press.

[WS01] I. Wald and P. Slusallek. State-of-the-art in interactive raytracing. In *Eurographics State of the Art Reports*, pages 21–42, 2001.

[WTP01] Alexander Wilkie, Robert F. Tobler, and Werner Purgathofer. Combined rendering of polarization and fluorescence effects. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 197–204, London, UK, 2001. Springer-Verlag.

[YCK92] R. Yagel, D. Cohen, and A. Kaufman. Discrete ray tracing. *IEEE Computer Graphics & Applications*, 12(9) :19–28, 1992.

[YIY97] Yizhou Yu, Oscar H. Ibarra, and Tao Yang. Parallel progressive radiosity with adaptive meshing. *Journal of Parallel and Distributed Computing*, 42(1) :30–41, 1997.

[ZCFM06] R. Zrour, P. Chatelier, F. Feschet, and R. Malgouyres. Parallelization of a discrete radiosity method. In W.E. Nagel, W.V. Walter, and W. Lehner, editors, *12th International Euro-Par Conference*, volume 4128 of *Lecture Notes in Computer Science*, pages 740–750. Springer-Verlag, 2006. to appear.

# Index

## Résumé

### Une approche de la radiosité par voxels, application à la synthèse d'images

La radiosité dans le domaine de la synthèse d'images est une technique de calcul d'éclairage. Elle est la solution d'une équation simplifiée par rapport au problème plus général de l'illumination globale. Plusieurs techniques algorithmiques existent pour résoudre cette équation, utilisant le lancer de rayons, ou la notion de facteurs de forme et une discrétisation de la scène en éléments surfaciques. Un algorithme inédit est présenté, utilisant une représentation à base de voxels et des notions mathématiques de géométrie discrète pour représenter différemment et résoudre plus efficacement le problème de la visibilité des éléments entre eux. Pour cela, un usage est fait de la notion de droites 3D discrètes. Cet algorithme présente une complexité quasi linéaire en temps et linéaire en espace. Une nouvelle représentation des BRDFs et des distributions de radiance a aussi été développée pour appliquer l'algorithme à la résolution de l'illumination globale plutôt que la simple radiosité. Elle est basée sur une interpolation de points de contrôles plutôt que sur une décomposition sur une base de fonctions. Le champ d'action de l'algorithme peut être étendu à la résolution du transfert radiatif en météorologie. Des améliorations sont également possibles dans le domaine de la parallélisation, l'algorithme se prêtant bien à une subdivision en tâches concurrentes et à la répartition des données. La brique de calcul développée doit maintenant être insérée dans un logiciel spécialisé permettant d'utiliser des méthodes mixtes et mieux contrôler les paramètres de l'algorithme permettant d'obtenir une image de qualité en un temps raisonnable.



**Mots-clefs :** Synthèse d'images, Radiosité, Illumination globale, Géométrie discrète, Droites 3D discrètes, voxels, algorithme quasi linéaire, BRDF, distributions de radiance, transfert radiatif, parallélisation.


---


## Abstract

### A voxel-based radiosity method, application to image synthesis

Radiosity in image synthesis is a way to compute lighting. It is the solution of an equation, which is itself the simplification of the problem known as global illumination. Several algorithms exist to solve this equation, using ray-tracing, or a notion of form-factors attached with the subdivision of the scene into patches. A new algorithm has been developed, using a voxel-based representation, and some mathematical notion from digital geometry, allowing to model differently, and compute more efficiently, the information of "visibility". It is based on the notion of 3D discrete lines. This algorithm has a quasi-linear time complexity, and a linear space complexity. A new representation of BRDFs, and of the derivated radiance distributions, has also been developed, so that the algorithm can be extended to the case of global illumination instead of mere radiosity. It is based on the interpolation of control points rather than decomposition over a basis of functions. The algorithm can also be used to solve the radiative transfert problem in meteorology. Improvements can also be made from the parallelization point of view, since the algorithm is well suited at being decomposed into several concurrent tasks and for data dispatching. The computation kernel now needs to be placed into a particular software allowing to use mixed methods and have a better control over the parameters of the algorithm, so that it is easier to tune the compromise between rendering quality and computation time.



**Keywords :** Image synthesis, Radiosity, Global Illumination, Digital Geometry, 3D discrete lines, Voxels, quasi-linear algorithm, BRDF, radiance distributions, radiative transfert, parallelization.