Université Paul Sabatier – Toulouse 3 **École Doctorale Informatique et Télécommunications**

Rendu, illumination et modèles à base de points

MÉMOIRE

présenté et soutenu publiquement le 27 Juin 2006

pour l'obtention du

Master Recherche

(Spécialité Image, Information Hypermédia)

par

Nicolas BONNEEL

Composition du jury

Rapporteur : Julien Pinquier

Encadrant : Mathias Paulin

Institut de Recherche en Informatique de Toulouse — UMR CNRS 5505

Remerciements

Je remercie particulièrement Mathias Paulin ainsi que Gaël Guennebaud qui m'ont tous deux grandement aidé par les heures d'explications du sujet et leur implication dans ce projet.

Table des matières

Table des figures

1

Présentation du sujet

Le but du stage est de calculer efficacement l'éclairage global dans un moteur de rendu temps-réel à base de points. On définira arbitrairement comme temps-réel un affichage à plus de 10 images par seconde, et comme temps interactif un affichage entre 2 et 10 images par seconde. Il s'agit ici de ne pas se limiter aux matériaux diffus ou "glossy" (peu spéculaires ou à la spécularité basse fréquence), mais de prendre en compte les matériaux dont la réflectance possède de très hautes fréquences afin de pouvoir rendre les matériaux très spéculaires.

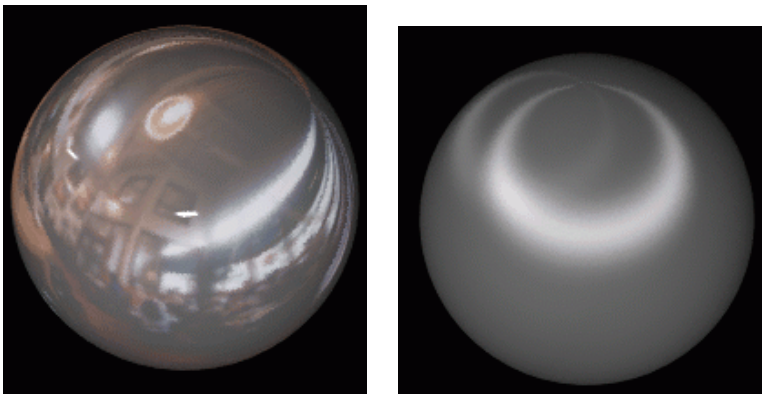

FIG. 1.1 – Sphère hautement spéculaire (gauche) et glossy (droite)

Le rendu doit être fait en temps réel, et le précalcul doit être le plus rapide possible.
Il s'agit donc de "Precomputed Radiance Transfer" ou PRT.

1.1 Équation du rendu

Le calcul de l'éclairage perçu dans une direction $\vec{\omega_r}$ en un point x_0 est donné par :

$$L(x_0, \vec{\omega_r}, \lambda) = L_e(x_0, \vec{\omega_r}, \lambda) + \int_{\Omega_i} L(x_0, \vec{\omega_i}, \lambda)\rho(x_0, \vec{\omega_r}, \vec{\omega_i}, \lambda)\vec{N_{x_0}}.\vec{\omega_i}d\vec{\omega_i} \tag{1.1}$$

où L est la luminance dans une direction donnée, L_e l'émissivité du matériau, λ la longueur d'onde, ρ la BRDF du matériau (voir plus loin), $\vec{N}$ la normale, et les indices i et r dénotent

respectivement les directions incidentes et réfléchies.
Il s'agit donc d'une intégrale récursive de Fredholm puisque la fonction à calculer $L(\omega)$ se retrouve aussi sous le signe intégrale.

Le terme $\vec{N_{x_0}}.\vec{\omega_i}$ tient compte de l'angle solide sous lequel la source lumineuse voit la surface éclairée.

On simplifiera cette équation en :

$$L(\vec{\omega_r}) = \int_{\Omega_i} L(\vec{\omega_i})\rho(\vec{\omega_r}, \vec{\omega_i})\vec{N}.\vec{\omega_i}d\vec{\omega_i} \tag{1.2}$$

en considérant les matériaux comme non émissifs (+ des sources lumineuses uniquement émissives), et en omettant les variables λ et x_0.

Cette formulation, bien qu'elle soit la plus utilisée, ne tient pas compte des effets tels que la fluorescence ou la phosphorescence.

1.2 L'éclairage global interactif

Il s'agit d'évaluer l'équation 1.2 pour ainsi prendre en compte les effets lumineux principaux nécessaires pour obtenir un résultat visuel plausible, ou même un résultat physiquement réaliste. En effet, cette équation étant difficile à évaluer, certaines approximations sont effectuées. Par exemple, on peut ne prendre en compte que l'éclairage direct (réduisant l'intégrale récursive en intégrale non récursive) ou alors que des matériaux diffus (l'éclairement ne dépend alors plus de $\vec{\omega_r}$ et peut donc se précalculer entièrement une fois pour toute) ou basses fréquences. On peut aussi ne considérer que des sources ponctuelles (on obtient alors des "ombres franches", mais l'intégration sur les sources lumineuses surfaciques émissives se réduit à une somme sur les sources lumineuses ponctuelles). Il s'agit donc d'effectuer le moins possible d'approximations afin de simuler un maximum d'effets physiques.

Fig. 1.2 – Comparaison éclairage direct/éclairage global (crédit : Nick Chapman)

1.3 *Bidirectional Reflectance Distribution Function (BRDF)*

1.3.1 Description

La fonction $\rho(\vec{\omega_r}, \vec{\omega_i})$ de l'équation 1.2, dans le cas où Ω_i et Ω_r sont des demi-espaces, est appelée Fonction de Distribution de la Réflectance Bidirectionnelle ou BRDF. Dans le cas plus général où Ω_i et Ω_r sont des espaces entiers ($\mathbb{R}^2$ tout entier), on parle de Bidirectional Surface Scattering Distribution Function ou BSSRDF.
On se limitera ici aux BRDF, qui ne peuvent donc pas prendre en compte les effets de diffusion sous-surfacique (effet fondamental dans le rendu de lait, de marbre, de peau etc...).

FIG. 1.3 – Comparaison d'un rendu de peau avec BRDF (gauche) et BSSRDF (droite) ([JMLH01])

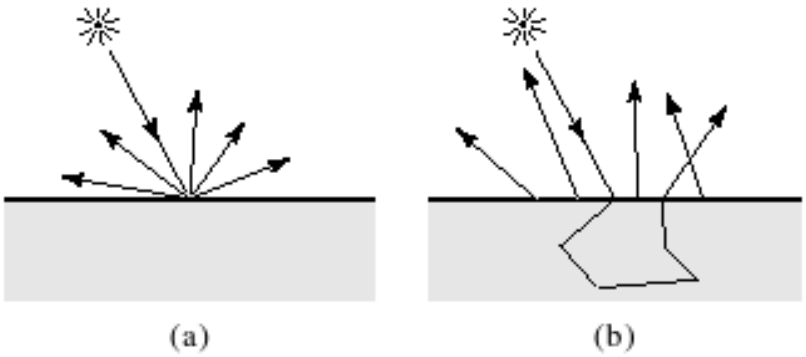

FIG. 1.4 – Diffusions prises en compte par les BRDF (a) et par les BSSRDF (b)

La BRDF donne le rapport entre l'énergie réfléchie dans la direction $\vec{\omega_r}$ d'un rayon lumineux issu de la direction $\vec{\omega_i}$. C'est donc une fonction spectrale d'une partie de $\mathbb{R}^2 \times \mathbb{R}^2$ dans $\mathbb{R}$ qui décrit comment la lumière est réfléchie par un matériau (la variable λ, la longueur d'onde, est omise par souci de lisibilité). Elle décrit les propriétés du matériau et contribue de manière importante au réalisme de la scène.

Physiquement, son expression est :

$$\rho(\vec{\omega_r}, \vec{\omega_i}) = \frac{dL_r(\vec{\omega}_r)}{dL_i(\vec{\omega}_i)}$$

où L_i est la radiance spectrale incidente (ou irradiance) et L_r la radiance spectrale sortante, à savoir les intensités lumineuses sur les éléments d'angles solides $d\Omega_i$ et $d\Omega_r$.

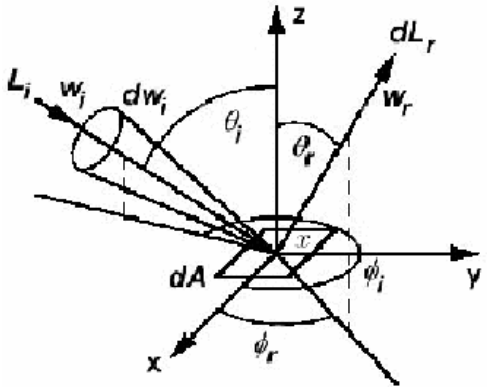

FIG. 1.5 – Notations pour les BRDF

Afin de simplifier l'équation 1.2, les BRDF sont souvent décomposées en deux parties : une partie "diffuse" ne comportant aucun terme fonction de $\vec{\omega_r}$, et une partie "spéculaire" comportant cette partie directionnelle. Ainsi, l'équation 1.2 se décompose en :

$$L(\omega_r) = \int_{\Omega_i} L(\vec{\omega_i})\rho_d(\vec{\omega_i})\vec{N}.\vec{\omega_i}d\vec{\omega_i} + \int_{\Omega_i} L(\vec{\omega_i})\rho_s(\vec{\omega_r}, \vec{\omega_i})\vec{N}.\vec{\omega_i}d\vec{\omega_i} \tag{1.3}$$

La partie $\int_{\Omega_i} L(\vec{\omega_i})\rho_d(\vec{\omega_i})\vec{N}.\vec{\omega_i}d\vec{\omega_i}$ de l'équation peut ainsi être calculée efficacement et simplement, une fois pour toute avant le rendu de la scène et être stockée comme une texture, permettant ainsi de faire varier le point de vue et de se mouvoir dans la scène sans calculs supplémentaires. La partie diffuse des scènes statiques est donc actuellement bien gérée, et c'est la partie spéculaire qui pose de nombreux problèmes en rendu interactif. Une décomposition diffus/spéculaire est montrée sur la figure 1.6 ainsi que l'image résultante.

Les BRDF doivent respecter des lois physiques :

– Principe de réciprocité d'Helmoltz :

$$f(\vec{\omega_r}, \vec{\omega_i}) = f(\vec{\omega_i}, \vec{\omega_r})$$

Ce principe énonce que l'intensité lumineuse perçue dans une direction $\vec{\omega_r}$ par un rayon réfléchi issu de la direction $\vec{\omega_i}$ est la même que celle perçue dans la direction $\vec{\omega_i}$ si le rayon provenait de la direction $\vec{\omega_r}$.

– Conservation de l'énergie :

$$\frac{1}{\pi}\int_{\Omega_i}\int_{\Omega_r} f(\vec{\omega_r}, \vec{\omega_i})\cos(\theta_r)d\vec{\omega_r}\cos(\theta_i)d\vec{\omega_i} \leq 1$$

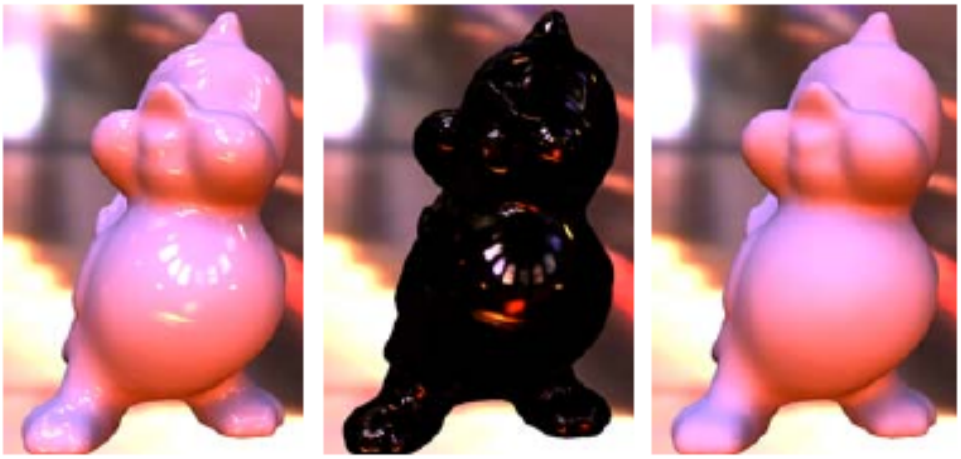

FIG. 1.6 – Image finale, partie spéculaire et partie diffuse ([GKMD06])

Ce principe énonce que matériau ne peut pas réfléchir plus d'énergie qu'il n'en a reçu.

La violation de ces lois entraîne un rendu physiquement irréaliste et des images non naturelles. Certaines BRDF ne respectent pas ces lois mais sont plus rapides à évaluer : si l'objectif n'est pas un rendu physiquement réaliste, l'utilisation de ces BRDF est tout à fait envisageable. Par ailleurs, beaucoup de BRDF sont basées sur une répartition en forme de lobe autour de la direction symétrique de $\vec{\omega_r}$ par rapport à la normale à la surface, propriété que nous utiliserons par la suite.

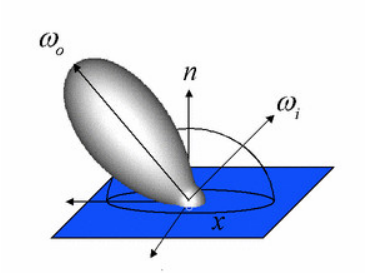

FIG. 1.7 – BRDF sous forme de lobe

1.3.2 Quelques BRDF

Il existe de très nombreux modèles de BRDF. Je ne décrirai ici que les principaux et ceux qui seront utilisés par la suite (les modèles à base de lobes essentiellement).

- Phong
  C'est la BRDF la plus courante par sa simplicité, mais, hormis les améliorations qui lui ont été apportées par [Lew94], ne respecte ni le principe de réciprocité, ni celui de conservation de l'énergie.

$$f(\vec{L},\vec{V}) = \frac{\rho_d}{2\pi} + \rho_s \left(\vec{R}.\vec{V}\right)^{n^P}$$

où n^P est l'exposant de Phong, et $\vec{R}$ le vecteur réfléchi de $\vec{L}$ (symétrique de $\vec{L}$ par rapport à la normale).

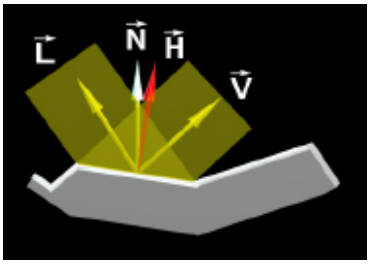

FIG. 1.8 – Notation des directions

- Blinn
  Une variante plus pratique à calculer consiste à utiliser le *half-vector* : $\vec{H} = \frac{\vec{L}+\vec{V}}{\|\vec{L}+\vec{V}\|}$. Ce modèle n'est pas lui non plus physiquement réaliste.

$$f(\vec{L},\vec{V}) = \frac{\rho_d}{2\pi} + \rho_s \left(\vec{H}.\vec{N}\right)^{n^B}$$

- Lewis
  C'est une version généralisée du modèle de Blinn en considérant une somme de BRDF de Blinn d'exposants différents.

$$f(\vec{L},\vec{V}) = \frac{\rho_d}{2\pi} + \sum_i \rho_{s_i} \left(\vec{H}.\vec{N}\right)^{n_i^B}$$

- Cook-Torrance
  Elle considère une distribution statistique de microfacettes à la surface du matériau.

$$f(\vec{L},\vec{V}) = \frac{\rho_d}{2\pi} + \rho_s \frac{FDG}{\left(\vec{N}.\vec{L}\right)\left(\vec{N}.\vec{V}\right)}$$

  avec F le terme de Fresnel, D la fonction de distribution des microfacettes et G un terme d'atténuation géométrique.
  Cette BRDF est physiquement réaliste.

- Ward
  [War92] décrit un modèle à base d'un lobe centré sur la direction symétrique de $\vec{\omega}_r$ par rapport à la normale à la surface, mais prenant en compte la forte spécularité aux angles rasants : en effet, lorsque l'on regarde un matériau à incidence rasante, il réfléchit toute l'intensité incidente.
  Pour ne décrire que sa version isotrope, sa formule est donnée par :

$$f(\theta_i,\varphi_i,\theta_r,\phi_r) = \frac{\rho_d}{\pi} + \rho_s \frac{1}{\sqrt{\cos\theta_i \cos\theta_r}}.\frac{\exp[-\tan^2\delta/\alpha^2]}{4\pi\alpha^2}$$

où $\theta_i, \varphi_i, \theta_r, \phi_r$ représentent respectivement $\vec{\omega_i}$ et $\vec{\omega_r}$ en coordonnées sphériques où l'axe z est la normale à la surface, et δ est l'angle entre le *half-vector* et la normale.
Le terme de normalisation $\frac{1}{4\pi\alpha^2}$ est correct pour des α inférieurs à 0.2 (tant que le matériau reste spéculaire). Une version anisotrope est aussi disponible.

– Lafortune
[LFTG97] propose aussi un modèle à base de sommes de lobes.

$$f(\vec{L}, \vec{V}) = \sum_i [C_{x,i} L_x V_x + C_{y,i} L_y V_y + C_{z,i} L_z V_z]^{n_i}$$

où les $C_{x,i}$, $C_{y,i}$, $C_{z,i}$ et n_i dépendent du matériau.
Ce modèle peut ainsi prendre en compte la spécularité aux angles rasants.

En pratique, ces BRDF modélisent les réflections issues de équations de Snell-Descartes sur des matériaux rugueux dont la rugosité est représentée par une distribution de microfacettes à leur surface. Ainsi, les coefficients respectivement n^p, n^B, et n_i^B permettent de faire varier la rugosité, la fonction D du modèle de Cook-Torrance permet de décrire la fonction de distribution de ces microfacettes, et le modèle de Ward suppose une distribution gaussienne de ces microfacettes. Par ailleurs, [APS00] propose une méthode de création de BRDF physiquement réalistes à partir de distributions quelconques de microfacettes.

Un résumé plus visuel des matériaux obtenus avec ces différents modèles de BRDF lorsqu'ils sont ajustés sur des mesures physiques est donné dans les figures 1.9 et 1.10.

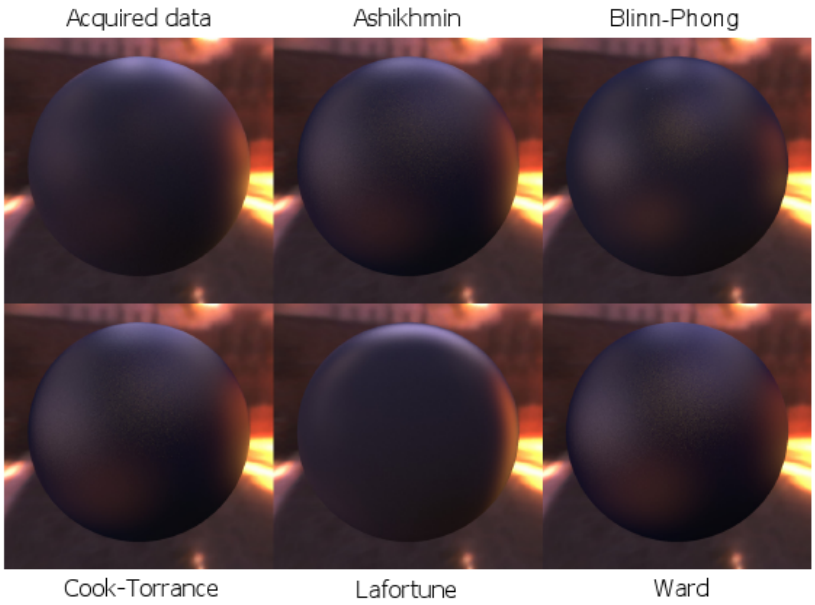

Fig. 1.9 – Peinture bleue acquise ajustée sur différents modèles de BRDF.

Les BRDF peuvent aussi provenir de mesures physiques obtenues grâce à des gonioréflectomètres. Pour une discrétisation de l'espace des directions incidentes, on obtient une distribution

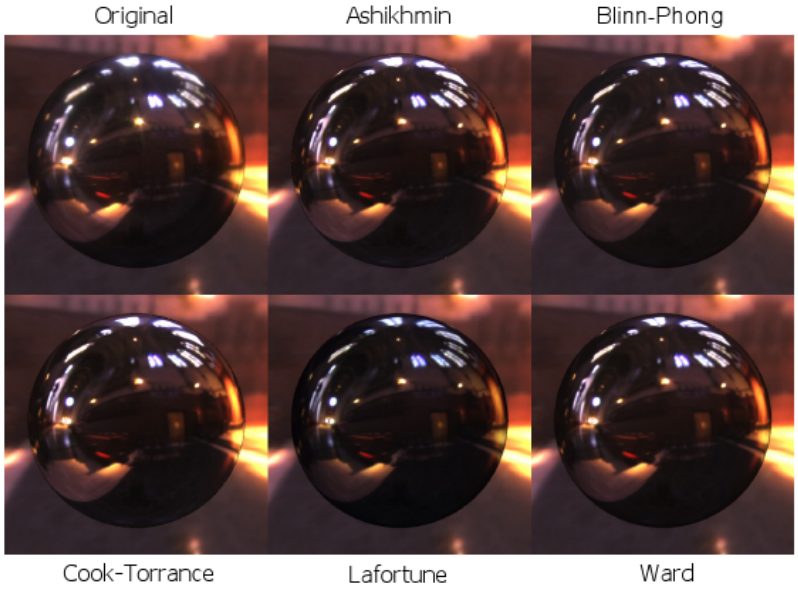

FIG. 1.10 – Chrome acquis ajusté sur différents modèles de BRDF.

d'énergie discrétisée pour les directions sortantes. En revanche, les BRDF issues de ces mesures ne sont pas paramétrables et correspondront donc à la réponse donnée par un matériau unique, mais le résultat sera plus réaliste.

1.3.3 Les matériaux *glossy* et spéculaires

Il est souvent mentionné dans les articles le terme de *glossy* pour des matériaux à la spécularité basse fréquence. En pratique, cette notion est très vague et englobe un peu toutes les spécularités qui ne sont pas vraiment directionnelles. Un aperçu de matériaux pouvant être considérés comme spéculaires et *glossy* est donné en figure 1.1.
Notre but est ici de pouvoir effectuer le rendu de matériaux spéculaires, et non *glossy*, à savoir des matériaux dont la BRDF possède de très hautes fréquences.

1.4 Méthodes de calcul de l'éclairage global

Les deux principales méthodes de calcul de l'éclairage global, à savoir la résolution de l'équation 1.2 sont le lancer de photons et la radiosité. Une méthode de Monte-Carlo bidirectionnelle est aussi utilisée, mais elle ne permet pas d'effectuer un précalcul de l'éclairage.

1.4.1 Lancer de photons

Il s'agit d'envoyer un flot de particules dans toutes les directions à partir des sources lumineuses et de les intersecter avec la scène. A chaque intersection, une direction de réflection est choisie aléatoirement en fonction de la BRDF du matériau (par échantillonnage par importance afin de réduire la variance et donc de pouvoir rendre au mieux les matériaux hautes fréquences)

et le photon est réémis dans cette direction (à moins qu'il n'ait été absorbé : ce choix se fait par un tirage aléatoire, l'algorithme de la roulette russe, qui détermine si le photon est réfléchi spéculairement, s'ils est diffusé ou s'il est absorbé). L'échantillonnage par importance se fait soit par la méthode d'inversion de la PDF (fonction de distribution de probabilité) cumulée d'une fonction la plus proche possible de la BRDF ou par la méthode de rejet suivant la BRDF. Une variance réduite permet d'éviter un bruit trop important dans la scène. A chaque réflection du photon, son énergie est multipliée par la BRDF du matériau et est divisée par la PDF utilisée pour générer la direction réfléchie, toutes deux estimées pour la direction incidente et réfléchie du photon.
On arrêtera la récursion grâce à l'algorithme de Monte-Carlo qui permet d'arrêter le processus en tirant un nombre aléatoirement. Les photons sont alors stockés à chaque intersection ainsi que leur direction incidente.
La densité de photons ainsi distribués est représentative de l'intensité lumineuse de la surface et permet de calculer une image réaliste en se servant de la BRDF pour évaluer l'intégrale de l'équation 1.2 qui devient non récursive. On peut alors effectuer un rendu de la scène à l'aide d'un lancer de rayon classique, en estimant la densité de photons à l'aide d'un noyau radial de convolution filtrant les photons stockés dans la phase précédente. [WJ95] a montré que le noyau optimal est le noyau d'Epanechnikov : $h(r) = \frac{3}{2}\left(1 - \frac{r^2}{R^2}\right)$. A titre de comparaison, le noyau constant a une efficacité théorique de 0.93 fois celle du noyau d'Epanechnikov.

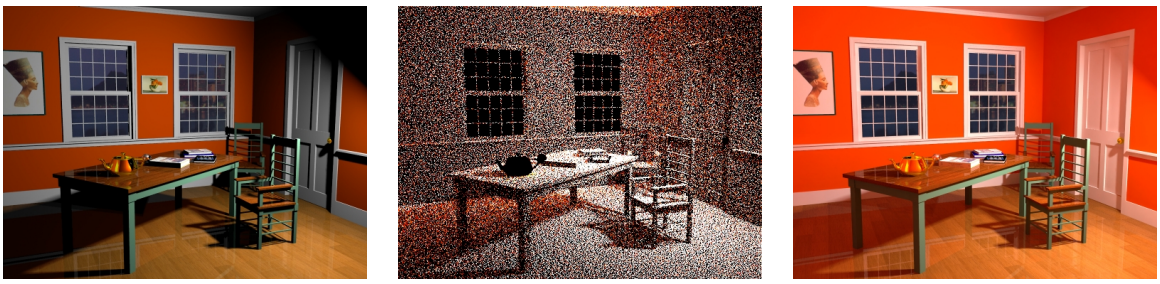

FIG. 1.11 – Éclairage direct, carte de photons et éclairage global (crédit Per H.Christensen)

1.4.2 Radiosité

Il s'agit juste de la résolution de l'équation 1.2 par éléments finis. Pour que cette résolution soit possible, il est nécessaire de considérer les matériaux comme étant entièrement diffus.
La discrétisation de l'équation résultante nous permet donc d'obtenir un système linéaire (généralement en considérant un noyau constant par triangle), souvent résolu par la méthode de Gauss-Seidel.
Sa discrétisation en éléments triangulaires rend cette méthode difficilement applicable au rendu par points, mais une solution a quand même été proposée en considérant les points comme des disques (voir état de l'art).

1.5 *Tone Mapping*

La résolution de l'équation 1.2 permet d'obtenir une intensité lumineuse en direction de l'observateur. Or, l'observateur, que l'on considérera humain, ne réagit pas linéairement en fonction de cette intensité. Ainsi, un réétalonnage linéaire des intensités obtenues entre 0 et 1 par exemple nous donnera une image à dominance sombre avec ponctuellement des intensités très fortes, ce qui ne correspond pas à la sensation visuelle humaine. Ce phénomène sera accentué pour les scènes hautement spéculaires que nous cherchons à rendre.
Idéalement, il faudrait utiliser les courbes de réponses de nos bâtonnets et cônes de notre oeil donnés par la CIE (Commission Internationale de l'Éclairage) et intégrer pour chaque longueur d'onde. En pratique, nous ne possédons souvent que 3 longueurs d'ondes : Rouge, Verte et Bleue. Il s'agit donc d'utiliser un opérateur, non linéaire si possible (voire dépendant du temps !), applicable en temps réel pour retrouver la perception visuelle humaine. C'est là trouver un opérateur de *Tone Mapping*.
Un exemple est fourni sur l'image 1.12 et une évaluation détaillée des opérateurs de Tone Mapping est disponible à [HLF].

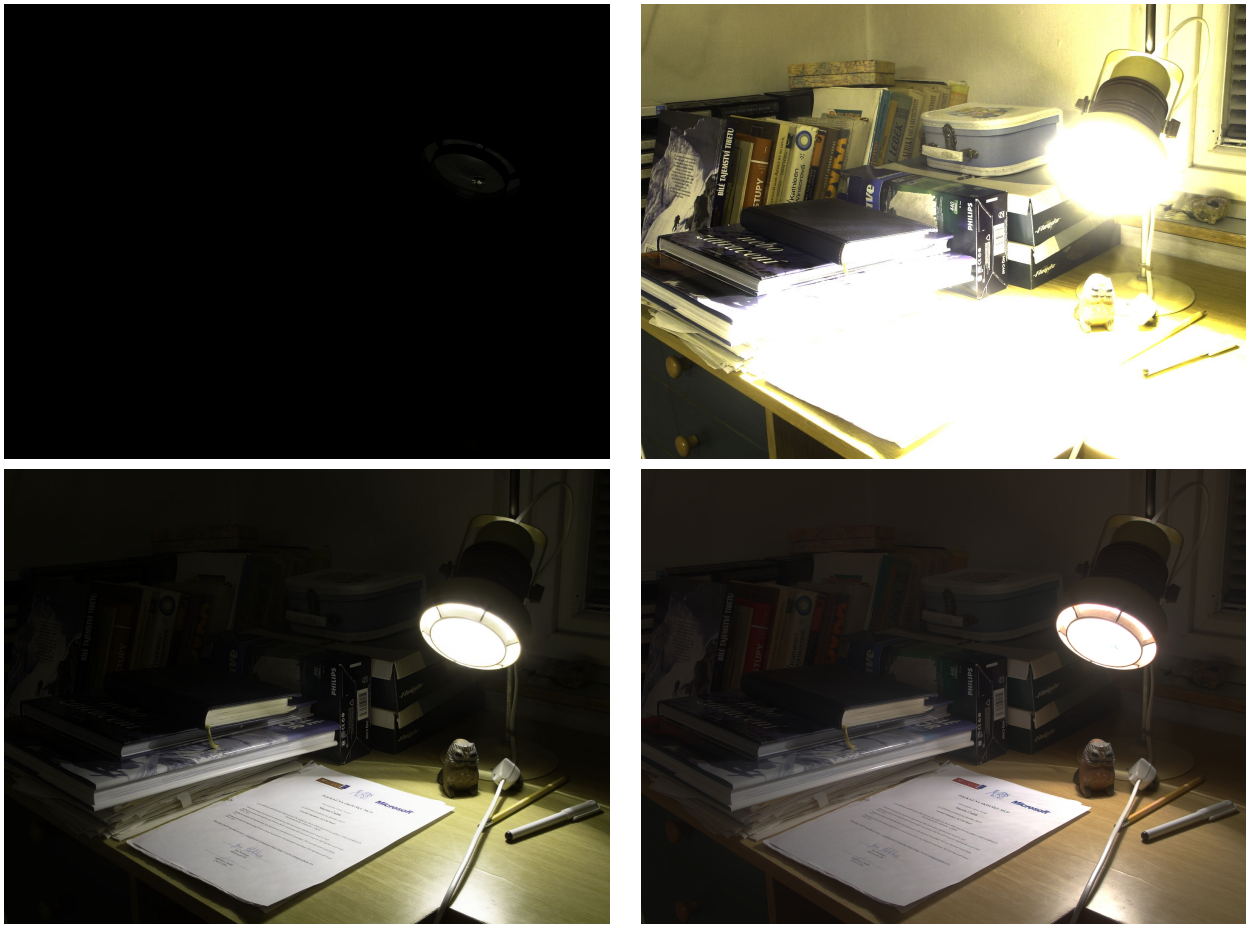

Fig. 1.12 – Comparaison mapping linéaire, clippé à 1.0, exponentiel, et selon la méthode de [TR93]

1.6 Rendu à base de points

La primitive usuelle pour l'affichage de modèles 3D est le polygone, et le plus souvent le triangle. Il se peut que le nombre de triangles soit si important qu'ils ne couvrent chacun que peu de pixels ou moins d'un seul pixel. Pour des modèles aussi détaillés, il est alors préférable d'utiliser un rendu où la primitive est le point, et chaque point est rendu comme une petite gaussienne couvrant quelques pixels, ce qui permet d'avoir un temps de rendu beaucoup plus court.
Un avantage de cette méthode est qu'elle évite le maillage du modèle (par exemple issu d'un scanner 3D) qui peut être très long, et le stockage de la topologie du modèle. Chaque point est ainsi considéré indépendamment, et possède ses propres caractéristiques (couleur et normale). La normale au point est évaluée localement sans avoir à mailler le modèle.

FIG. 1.13 – Forêt à base de points, scène typiquement difficile à afficher avec des polygones (G.Guennebaud, IRIT)

2

État de l'art

Dans cet état de l'art, nous allons nous concentrer sur les méthodes permettant le précalcul de l'éclairage global.
On effectue donc ce précalcul par un lancer de photons ([KAMJ05]), ou simplement en utilisant une carte d'environnement (ou "environment map" : une photographie d'un ciel par exemple, donnant ainsi la répartition spectrale de la lumière incidente lorsque la scène est éclairée par un ciel). Dans le premier cas, tous les effets lumineux peuvent être pris en compte, alors que dans le second, les sources sont supposées à l'infini. Une approche hybride étant de considérer une carte d'environnement en plusieurs points de la scène (une carte par objet, par point...) ce qui permet de considérer des effets lumineux plus locaux mais au détriment de l'occupation mémoire.

Une fois calculé, l'éclairage doit être stocké. Ce stockage pouvant être coûteux en mémoire (stocker les millions de photons et leur directions, ou bien stocker des cartes d'environnement en chaque point de la scène), plusieurs méthodes ont été proposées pour compresser ces données : décomposition dans une base (ondelettes, harmoniques sphériques, harmoniques zonales), analyse en composantes principales (ACP)...

2.1 Préambule : distributions sphériques

L'éclairage d'une scène peut être précalculé par lancer de photons ou par une carte d'environnement. Le lancer de photons permet, en envoyant des millions de photons, d'obtenir en chaque point de la scène une distribution incidente d'énergie sur la demi-sphère : pour chaque direction incidente $\vec{\omega}_i$ est associée une énergie $F(\vec{\omega}_i)$. De la même manière, une carte d'environnement stocke la distribution d'énergie issue de sources lumineuses supposées à l'infini. En pratique, la carte d'environnement est une *cubemap* : la distribution d'énergie provenant des directions projetées sur les faces d'un cube (voir illustration sur la figure 2.8), ou bien une carte sphérique représentant l'énergie provenant des directions directement projetées sur la demi-sphère.

De même, une BRDF fournit pour chaque direction incidente $\vec{\omega}_i$ une distribution sphérique de réflectance $\rho(\vec{\omega}_r, \vec{\omega}_i)$. Une représentation est donnée figure 1.7 montrant la répartition de l'énergie réfléchie en fonction du vecteur incident $\vec{\omega}_i$.

Nous allons voir dans cet état de l'art les méthodes permettant de stocker ou représenter efficacement ces distributions, ce qui permettra l'évaluation en temps réel de certaines modifi-

cations de la scène (déplacements de caméra, rotation ou déformation d'objets, changement de l'éclairage ou édition des matériaux). Le faible coût de stockage est important car le transfert des informations stockées vers la carte graphique peut être trop lent pour pouvoir visualiser des modifications de la scène interactivement.

2.2 Compression par ondelettes, harmoniques sphériques, harmoniques zonales, ...

Nous avons vu que le calcul de l'éclairage perçu dans une direction $\vec{\omega}_r$ en un point x_0 est donné par l'équation 1.2.

On pose $\sigma(\vec{\omega}_r, \vec{\omega}_i) = \rho(\vec{\omega}_r, \vec{\omega}_i)\vec{N}.\vec{\omega_i}$. Lorsqu'on applique un algorithme de lancer de photons ou que l'on utilise une carte d'environnement, nous connaissons l'énergie incidente en tout point de la scène, donnée par une fonction $F(\vec{\omega}_i)$ ce qui permet de supprimer la récursivité de l'intégrale à calculer.
On obtient alors :

$$L(\vec{\omega}_r) = \int_{\Omega_i} F(\vec{\omega}_i)\sigma(\vec{\omega}_r, \vec{\omega}_i)d\vec{\omega}_i$$

Un point fort de la décomposition de la fonction F et de ρ dans une base orthonormée quelconque est que cette intégrale (double, car nous sommons sur toutes les directions) se résume au produit scalaire des composantes de F et de σ, car :

$$\begin{cases} F(\vec{\omega}_i) & = & \sum F_i e_i^1(\vec{\omega}_i) \\ \sigma(\vec{\omega}_r, \vec{\omega}_i) & = & \sum \sigma_i(\vec{\omega}_r) e_i^2(\vec{\omega}_i) \\ L(\vec{\omega}_r) & = & \int_{\Omega_i} \sum_i \sum_j \left(\sigma_i(\vec{\omega}_r) e_i^2(\vec{\omega}_i)\right)\left(F_j e_j^1(\vec{\omega}_i)\right) \end{cases}$$

où les F_i (resp. $\sigma_i(\vec{\omega}_r)$) sont les coefficients de la projection de la fonction $F(\vec{\omega}_i)$ (resp. $\sigma(\vec{\omega}_r, \vec{\omega}_i)$) dans la base des e_i^1 (resp. e_i^2). Attention, $\vec{\omega}_i$ dénote le vecteur incident et non un vecteur fonction de l'indice i. Ce système est alors équivalent à :

$$\begin{cases} F(\vec{\omega}_i) & = & \sum F_i e_i^1(\vec{\omega}_i) \\ \sigma(\vec{\omega}_r, \vec{\omega}_i) & = & \sum \sigma_i(\vec{\omega}_r) e_i^2(\vec{\omega}_i) \\ L(\vec{\omega}_r) & = & \sum_i \sum_j \sigma_i(\vec{\omega}_r) F_j \int_{\Omega_i} e_i^2(\vec{\omega}_i) e_j^1(\vec{\omega}_i) \end{cases}$$

Or, si la base utilisée est orthonormée, on a par définition :

$$\int_{\Omega_i} e_i^2(\vec{\omega}_i) e_j^1(\vec{\omega}_i) = \delta_{i,j}$$

avec $\delta_{i,j}$ le symbole de Kronecker ($\delta_{i,j} = 1$ ssi $i = j$, $\delta_{i,j} = 0$ sinon).
Et l'intégrale souhaitée se calcule alors simplement par :

$$L(\vec{\omega}_r) = \sum_i \sigma_i(\vec{\omega}_r) F_i$$

On voit alors directement l'intérêt de décomposer la BRDF et l'énergie incidente dans une base orthonormée afin de calculer l'équation de rendu par un simple produit scalaire.

Une extension pour le cas de l'intégrale du triple produit a été proposée par [NRH04] mais nécessite le stockage des *tripling coefficients* : $\int_{\Omega_i} e_i^1(\vec{\omega}_i)e_j^2(\vec{\omega}_i)e_k^3(\vec{\omega}_i)d\vec{\omega}_i$, et une triple sommation des coefficients en $\mathcal{O}(n^3)$ qui pour certaines bases peut se faire en $\mathcal{O}(n\log(n))$ (la base d'ondelettes de Haar) avec n le nombre de coefficients souhaités.

Les bases les plus couramment utilisées sont les bases d'ondelettes de Haar planes ([NRH03, WTL04]) ou sphériques ([Cla03]), les harmoniques sphériques ([SKS02, Gre03]), les harmoniques zonales ([SLS05]) qui sont un cas particulier des harmoniques sphériques et les polynômes de Zernike. Par ailleurs, [GKPB04] a proposé une base orthonormée basée sur les polynômes de Legendre tout comme les harmoniques sphériques, mais définie sur l'hémisphère. De plus, les polynômes de Jacobi ont été utilisés dans le cadre du transfert radiatif ([GKPB04]). Plus récemment, la base de Daubechies 4 (plane) a été utilisée par [BAOR06], et les bases de fonctions radiales sphériques par [TS06].

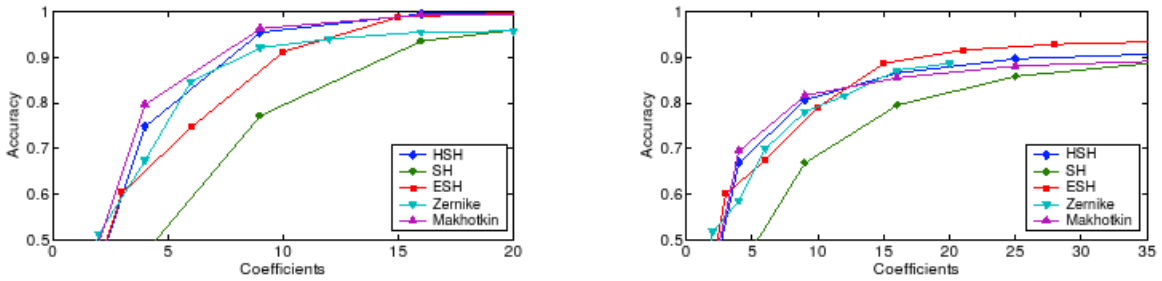

FIG. 2.1 – Comparaison des bases les plus courantes (harmoniques hémisphériques de [GKPB04], harmoniques sphériques, harmoniques sphériques paires de [SHHS03], les polynômes de Zernike, et la base de Makhotkin issue des polynômes de Jacobi) pour l'approximation d'une lobe de Phong d'exposant 5 à gauche et d'un lobe de Ward anisotrope ($\alpha_x = 0.2, \alpha_y = 0.5, \rho_s = 1$) à droite. Tracé de la précision en fonction du nombre de coefficients pour approximer. ([GKPB04]). A noter que d'après [WTL04], l'erreur sur l'image rendue n'a pas de liaison directe avec l'erreur sur la BRDF.

Nous présentons ici les principales bases utilisées pour représenter ces distributions sphériques. Toutes ces bases sont orthonormées, et donc bénéficient de cette approche d'évaluation de l'équation du rendu par un produit scalaire.

2.2.1 Ondelettes

La base d'ondelettes de Haar est très utilisée par la simplicité de décomposition des fonctions dans celle-ci, ainsi que la compression des coefficients. Elle a beaucoup été utilisée dans le but de rendre des matériaux glossy car elle offre une très bonne localisation spatiale. La base de Daubechies 4 est récemment apparu dans le domaine des PRT avec les travaux de [BAOR06] mais reste plus complexe.

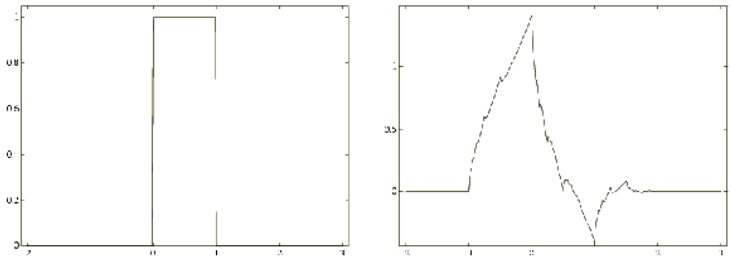

FIG. 2.2 – Bases d'ondelettes de Haar (gauche) et de Daubechies 4 (droite) en 1D

Ainsi, [NRH03] stocke les coefficients de la transformée en ondelettes des faces d'un cubemap dans la base de Haar en 2D. Le précalcul est de 3h pour un modèle de 80 000 sommets, le rendu est interactif, et l'approche par cubemap nécessite des sources supposées distantes. Selon eux, moins de 1% des coefficients sont necessaires pour obtenir une bonne qualité visuelle et ils proposent de garder entre 40 et 200 coefficients optimaux sur 24 576, mais se limitent aux scènes principalement diffuses. La compression est aisée car il suffit de supprimer tous les coefficients inférieurs à un certain seuil (ou de ne garder que les N plus grand coefficients).
L'argument présenté pour utiliser la base d'ondelettes est qu'elles ont une meilleure localisation en espace que les harmoniques sphériques. Le résultat est donné en figure 2.3. Nous voyons que le résultat est en effet peu spéculaire pour un temps de calcul et un volume de stockage importants. Par ailleurs, [DLWA] montre le manque de localisation en fréquence de la base de Haar par son manque de régularité. De plus, [GKPB04] montre les problèmes d'aliasing lorsque des fonctions lisses sont approximées avec un nombre trop peu important d'ondelettes (utilisation d'ondelettes sphériques pour la représentation des BRDF dans leur papier). Ils présentent aussi le fait que l'aliasing que cela provoque affecte la qualité visuelle d'une manière plus importante qu'une reconstruction plus lisse obtenue avec des harmoniques sphériques par exemple.

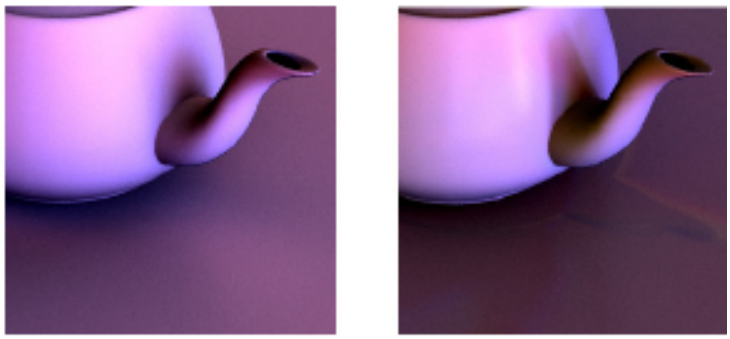

FIG. 2.3 – Théières glossy rendues : à gauche avec 200 harmoniques sphériques linéaires (1.3% d'erreur $\mathcal{L}^2$) , à droite avec 200 ondelettes de Haar non linéaires (0.46% d'erreur $\mathcal{L}^2$) ([NRH03]).

D'une manière analogue, [BAOR06] a très récemment proposé une décomposition d'une paramétrisation 1D de la BRDF dans une base d'ondelettes de Daubechies 4 en considérant le point de vue fixé. La base d'ondelettes de Daubechies 4 possède l'avantage d'être beaucoup plus lisse que la base d'ondelettes de Haar. De plus, la décomposition d'une fonction 1D plutôt que 2D (le point de vue étant fixé, la BRDF se restreint à 2 dimensions, mais une reparamétrisation

1D de la BRDF, possible pour la plupart des BRDF, est effectuée) permet de stocker beaucoup moins de coefficients pour un résultat plus précis. Ils stockent ainsi jusque 256 coefficients pour un rendu très proche des images de référence. Cette approche est similaire aux précédentes puisque l'équation du rendu se résume simplement à un produit scalaire entre les coefficients de la BRDF et ceux de la carte d'environnement avec la prise en compte de sa visibilité, la différence résidant dans le fait que c'est le point de vue qui est fixé et non la BRDF. Cette méthode leur permet d'éditer en temps réel des BRDF à l'aide de courbes 1D représentant divers paramètres. La carte d'environnement est discrétisée en patches triangulaires permettant un rendu plus précis qu'une discrétisation ponctuelle. Un temps de précalcul d'une quinzaine de minutes est nécessaire afin de déterminer les triangles visibles de la carte d'environnement en chaque pixel (pour une image résultante de 512x512 pixels) par une méthode de lancer de rayons classique et de projeter leur énergie sur la base de Daubechies 4 en 1D. Par ailleurs, ils utilisent la cohérence temporelle entre les images pour ne calculer qu'un nombre restreint de termes du produit scalaire de l'équation du rendu, et de mettre à jour progressivement l'image pour un résultat plus précis. Ainsi, seuls une vingtaine de termes sont calculés pour le produit scalaire pour chaque pixel. Ce calcul est effectué par la carte graphique, et est donc suffisamment rapide pour être effectué en temps réel (25 images par seconde). En revanche, les inter-réflections ne sont pas gérés. Les résultats sont montrés en figure 2.4.

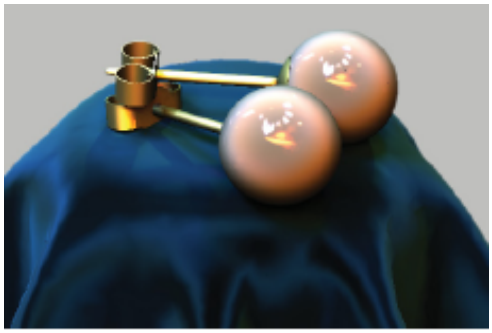

Fig. 2.4 – Scène où la mise à jour des matériaux est effectuée en temps réel après 15.3 min de précalcul ([BAOR06]).

2.2.2 Harmoniques sphériques

La base d'harmoniques sphériques est une base construite sur les polynômes de Legendre. Elle a beaucoup été utilisée par sa simplicité et ses propriétés d'invariance par rotation qui évite les effets d'aliasing quand la fonction est reconstruite à partir d'un ensemble de points ayant subit une rotation. Une interpolation précise de deux fonctions projetées sur la base d'harmoniques sphériques est rendu possible grâce au calcul du gradient de translation et de rotation entre les points de la surface. En revanche, la compression des coefficients est très coûteuse puisqu'une étape d'optimisation non linéaire est à appliquer, plus complexe que pour la projection dans une base d'ondelettes.

[SKS02] utilise une base d'harmoniques sphériques pour stocker de 9 à 25 coefficients d'une carte d'environnement. On applique alors une série d'opérateurs dont les fonctions de transfert sont données. Ils se limitent aux matériaux à basse fréquences, et disent pouvoir gérer des matériaux "glossy" en conservant plus de coefficients, mais admettent qu'ils ne pourront jamais

gérer de matériaux hautement spéculaires avec cette méthode. Cette méthode permet d'opérer à des rotations de l'objet. Leur méthode permet aussi de gérer le rendu volumique, ce qui peut être utile pour le rendu de scènes avec milieu participant (fumée, brouillard). En revanche, l'emploi d'une carte d'environnement est peu intuitive comparé au positionnement de sources lumineuses réelles dans la scène. Un temps de précalcul de 8 min pour les scènes diffuses à plus de 4h pour les modèles glossy (théière avec reflexions multiples) est nécessaire sur un P4 2.2GHz.

FIG. 2.5 – Un bouddha glossy (2h30 de précalcul, 50 000 sommets). ([SKS02]).

[Gre03] utilise aussi une base d'harmoniques sphériques mais pour stocker l'illumination issue d'un lancer de photons avec 16 coefficients. Cette méthode ne marche qu'avec des modèles statiques mais ils espèrent pouvoir améliorer leur méthode pour gérer les modèles dynamiques au prix de quelques précalculs en plus. Ils ne s'occupent que de matériaux principalement diffus.

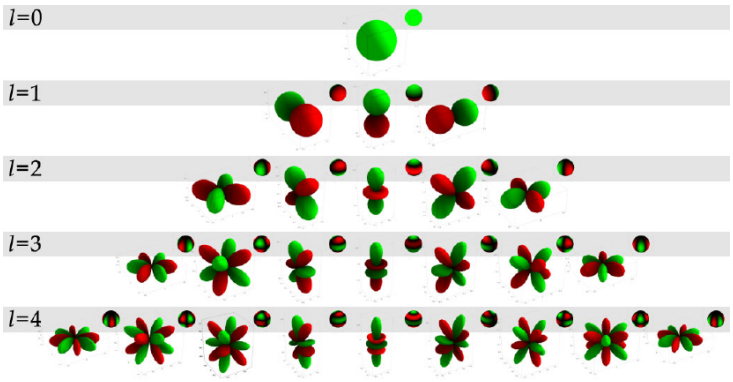

FIG. 2.6 – Illustration des harmoniques sphériques de l'ordre 0 à 4 (positif en vert, négatif en rouge) ([Gre03]).

FIG. 2.7 – Voiture rendue à partir d'un modèle à base d'harmoniques sphériques ([Gre03]).

[DLWA] montre aussi que les harmoniques sphériques ne peuvent avoir de fréquences assez élevées qu'à partir de 1000 à 3000 coefficients (harmoniques d'ordre 50 environ). Le stockage d'un tel nombre de coefficients pour chaque point de la scène est beaucoup trop coûteuse en mémoire, et le produit scalaire lors du rendu est beaucoup trop long à évaluer.

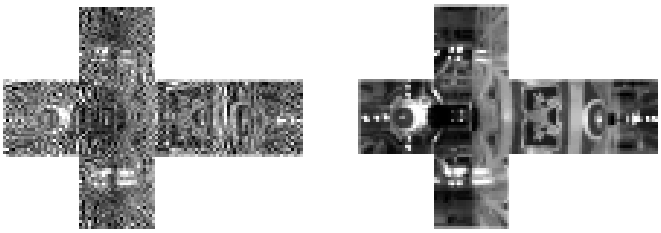

FIG. 2.8 – Cubemaps compressées avec : 10 000 harmoniques sphériques à gauche (48% d'erreur) , 4096 ondelettes de Haar non linéaires à droite (0.6% d'erreur) ([NRH03]).

2.2.3 Harmoniques hémisphériques

Remarquant que représenter des fonctions hémisphériques sur une base sphérique nécessitait de conserver de nombreux coefficients inutiles, [GKPB04] a proposé une base de fonctions hémisphériques basée aussi sur les polynômes de Legendre. Le stockage est ainsi plus compact et la reconstruction est plus précise et presque aussi rapide que la reconstruction sur une base d'harmoniques sphériques.
Cette décomposition est appliquée aux BRDF et à la radiance incidente issue d'un lancer de photons. L'approche par carte d'environnement est aussi évoquée, mais nécessite une pré-projection de la carte d'environnement dans une base d'harmoniques sphériques puis une transformation dans la base d'harmoniques sphériques grâce à une matrice de passage, effectuée après rotation des coefficients.
[GKPB04] signale que l'utilisation de cette base pour représenter des distributions hautes fré-

quences nécessiterait de trop nombreux termes.

2.2.4 Polynômes de Zernike et base de Makhotkin

Il s'agit d'une base de fonctions définies sur un disque. Cette base a été adaptée pour représenter des fonctions hémisphériques et représenter des BRDF mesurées.
Selon [GKPB04] l'évaluation de ces fonctions est très coûteuse, 10 fois plus que l'évaluation des polynômes de Legendre des harmoniques sphériques, et ne possède pas de matrices de rotation.

Par ailleurs [GKPB04] signale l'existence de la base de Makhotkin issue des polynômes de Jacobi, facile à évaluer par récurrence, mais qui ne possède pas de matrice de rotation non plus.

2.2.5 Fonctions radiales sphériques

Plus récemment (travaux publiés en fin de stage), [TS06] utilise une base de fonctions radiales sphériques gaussiennes ou d'Abel-Poisson pour représenter au mieux une carte d'environnement. Cette approche semble plus précise que l'utilisation d'ondelettes, mais nécessite toutefois l'utilisation de plusieurs centaines de coefficients. Le temps de précalcul est d'environ 2h pour le précalcul de l'éclairage, puis de plus de 20h pour la compression des coefficients, ce qui est prohibitif. Ils obtiennent ainsi le lapin présenté en image 2.10.

FIG. 2.9 – Référence puis cubemaps compressées avec : 972 harmoniques sphériques à gauche (96.06% d'erreur) , 972 ondelettes de Haar non linéaires ensuite (46.60% d'erreur) puis 972 fonctions radiales sphériques (3.14 % d'erreur) ([TS06]).

2.3 Local Deformable Precomputed Radiance Transfer (LDPRT)

L'utilisation d'une simplification des harmoniques sphériques : les harmoniques zonales, ont permis à [SLS05] de prendre en compte l'illumination globale sur des modèles animés par des transformations non rigides. Cette simplification leur permet, grâce à l'utilisation d'un à trois lobes (les harmoniques zonales prenant la forme de "lobes"), d'utiliser efficacement la propriété d'invariance par rotation des harmoniques sphériques.
Ils capturent ainsi des détails locaux comme la diffusion sous-surfacique, les inter-réflections diffuses et les surfaces "glossy" mais ne peuvent pas prendre en compte les effets distants comme les ombres d'un objet sur lui-même.
Le résultat est obtenu après un précalcul de moins de 30min sur un P4 3GHz, mais le rendu est temps-réel.

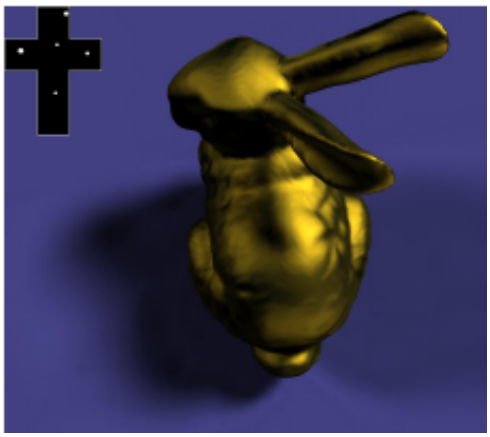

FIG. 2.10 – Lapin glossy affiché à 10.42 fps après 31h 49min de précalcul et compression pour 61 000 sommets ([TS06]).

Par ailleurs, l'utilisation des harmoniques zonales restreint la méthode aux matériaux peu spéculaires (voir l'illustration de ce qu'ils appellent "glossy" sur la figure 2.11).

FIG. 2.11 – Un vase dit "glossy" ([SLS05]).

2.4 Precomputed Local Radiance Transfer (PLRT)

Ainsi, Kristensen et al. proposent dans [KAMJ05] un modèle complet permettant de prendre en compte l'éclairage global de sources lumineuses ponctuelles et animées en considérant une grille contenant plusieurs positions de la source lumineuse. L'éclairage est ainsi calculé pour chaque position possible de la lumière sur toute la scène, et, étant donné la quantité de données à stocker, une première passe de projection sur une base d'harmonique sphériques est opérée, puis une séparation de la scène en zones où l'éclairage est similaire, puis une Analyse en Composantes Principales (ACP) afin de réduire le volume des données.
La phase de rendu se fait en interpolant les coefficients de la base entre les différentes positions où l'éclairage global a été calculé et la position réelle de la lumière. Le rendu est temps-réel.
Les résultats sont donc obtenus après un très long précalcul (plus de 2h pour une boîte de

Cornell), et sont limités aux sources ponctuelles. Une amélioration aux sources directionnelles est proposées en échange d'une augmentation du nombre de dimensions au problème. De plus, la décomposition en harmoniques sphériques limite la méthode aux matériaux dits "glossy".
Une image est donnée en figure 2.12.

FIG. 2.12 – Lapin rendu avec des harmoniques sphériques du 3ème ordre en ne conservant que 16 fonctions de base. 3000 triangles, 3h30 de précalcul avec la méthode de [KAMJ05] sur un cluster de 32 ordinateurs à 2.8 GHz.

2.5 Utilisation de BRDF séparable

Wang et al. utilisent dans [WTL04] une décomposition d'une matrice représentant la BRDF en vecteurs et valeurs propres afin de ne conserver que les plus grands vecteurs propres (les 4 premiers pour des modèles à spécularité modérée). Cette méthode considère la BRDF comme séparable, ce qui permet d'envisager dans un premier temps une décomposition en valeur singulières de celle-ci. Elle utilise ensuite le principe de réciprocité d'Helmoltz qui permet de symétriser la matrice et ainsi de simplifier le problème de plus grandes valeurs singulières en problème de plus grandes valeurs propres que l'on résout facilement et à faible coût par la méthode des puissances itérées. Ils construisent alors une *lightmap* à partir des vecteurs propres et une *viewmap* à partir des valeurs propres pour plusieurs directions réfléchies. La carte d'environnement est donnée par une cubemap compressée par ondelettes (base de Haar pour chaque face) en conservant 96 termes (par face).
Ils arrivent alors à effectuer un rendu temps-réel lorsque le point de vue change, et presque temps-réel lorsque l'éclairage change, les calculs étant accélérés car effectués sur GPU. L'éclairage est supposé distant. Ils arrivent alors à rendre des matériaux à plutôt forte spécularité. Un précalcul de près de 4h est nécessaire pour obtenir les 4 vecteurs propres principaux pour une scène de 200 000 sommets. Un résultat est donné en figure 2.13.

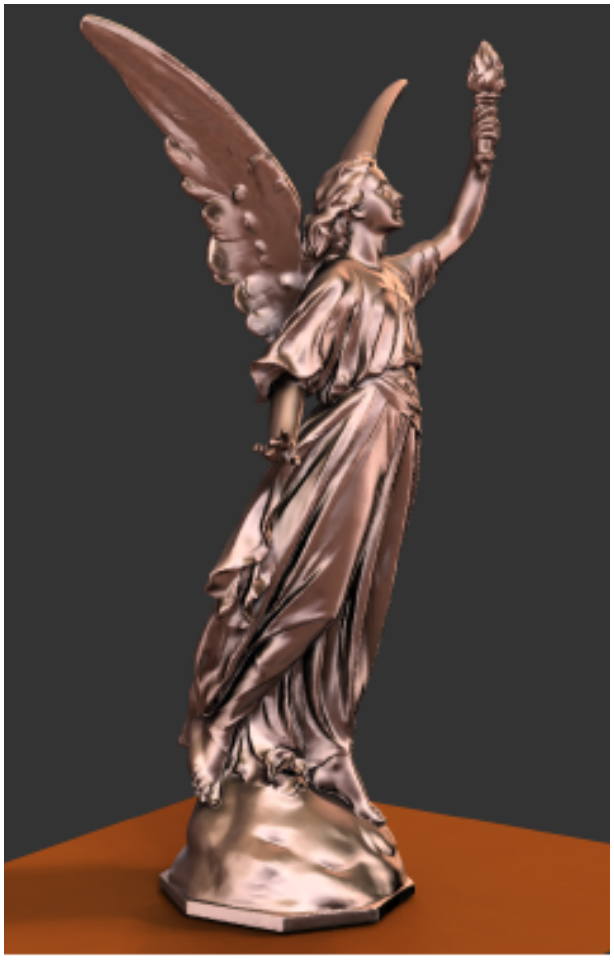

FIG. 2.13 – Lucy plutôt spéculaire rendue en conservant 4 vecteurs propres. 233 min de précalcul. 12fps lorsque le point de vue change, 1.7 fps lorsque l'éclairage change ([WTL04]).

2.6 Rendu par "sphère"

Une approche différente est de ne pas considérer directement l'équation du rendu, mais des méthodes qui aboutissent à des résultats visuellement acceptables sans se préoccuper de leur réalité physique.
Une méthode de ce type a été donnée par [IVD+04] : il s'agit de précalculer l'éclairage d'une sphère dont le matériau est le même que le matériau de l'objet à rendre, pour chaque lumière de la scène. On applique alors l'image de cette boule comme texture de réflection de l'objet afin de simuler les reflets spéculaires. Les reflets que nous obtenons sont juste issus de cette image précalculée.
Cette méthode, bien que très grossièrement approximative (voir figure 2.14), donne des résultats visuellement corrects pour des matériaux de type "peintures de voitures" mais est limitée à des sources distantes et des matériaux peu spéculaires. En revanche, cette méthode ne constitue pas une méthode d'éclairage global puisqu'elle ne prend pas en compte les inter-réflections etc... : il s'agit simplement d'un rendu plausible, mais non physiquement réaliste. Le calcul et le rendu sont entièrement temps-réel ce qui permet d'envisager le rendu de scènes dynamiques.

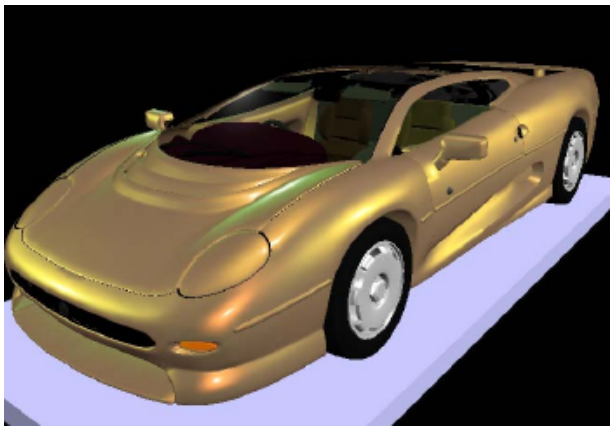

FIG. 2.14 – Voiture peinte rendue avec la méthode par "sphère"

2.7 Prefiltered Environment Maps et Mélange de gaussiennes

Une modélisation similaire à la nôtre, mais avec une approche différente a récemment été proposée par [GKMD06] : l'éclairage spéculaire direct est calculé par un fragment shader et l'éclairage spéculaire indirect est calculé par lancer de photon. Ils produisent ainsi des *cubemap* qui vont être approximés par un mélange de gaussiennes grâce à une optimisation non linéaire en chaque point de la scène. La composante diffuse est approchée par une base d'harmoniques sphériques à l'ordre 5.
L'éclairage est ainsi stockée pour 92 direction de la caméra dans des cartes d'environnement préfiltrées pour chaque gaussienne : en effet, comme le remarque [DHS+05], l'équation du rendu peut être vu comme convolution sous certaines conditions (nous y reviendrons par la suite). On applique donc une convolution à la carte d'environnement dont le noyau est chaque gaussienne, et nous obtenons ainsi plusieurs cartes d'environnement filtrées.
Le temps de précalcul est de 1h à 3h sur un cluster de 20 machines. La méthode est limitée aux sources lumineuses distantes mais permet de rendre des matériaux hautement spéculaires. Le temps de calcul est donc prohibitif mais il n'y a pas de limitation aux matériaux *glossy*.
Un terme favorisant la continuité des directions de chaque gaussienne a été ajouté. De cette manière, chaque gaussienne d'un sommet de la scène est similaire à la gaussienne de ses sommets voisins : deux gaussiennes ne pourront donc pas être inversées d'un sommet à l'autre. Ceci permet d'interpoler directement les directions des gaussiennes pour recalculer un éclairage précis entre les sommets, au lieu d'interpoler l'éclairage directement ce qui résulterait en une perte de résolution fréquentielle. Il s'agit de la même "astuce" qui nous ferait choisir un éclairage de Phong (qui interpole les normales, et qui permet donc de rendre des matériaux métalliques) au lieu d'un éclairage de Gouraud (qui interpole l'éclairage).
Un résultat fortement spéculaire est montré en figure 2.15.

La technique de carte d'environnement préfiltrée a précédemment été abordée par [KM00] pour des BRDF représentées par une somme de lobes quelconques invariants par symétrie radiale, en pratique de 1 lobe pour les BRDF très basses fréquences à 5 lobes pour les BRDF un peu

FIG. 2.15 – Oiseau très spéculaire rendu par mélange de gaussiennes ([GKMD06]).

plus directionnelles. Par ailleurs, ils déconseillent l'utilisation de lobes de Phong à cause de son invariance pour les angles rasants. Par ailleurs, l'utilisation de somme de 6 à 8 lobes augmente l'erreur commise sur l'approximation de la BRDF.

FIG. 2.16 – Théière avec une BRDF de Ward, avec 3 lobes ([KM00]).

2.8 Lancer de photons

Afin de calculer l'éclairage en temps réel sans précalcul et ainsi gérer des scènes dynamiques quelconques, une solution consiste à limiter la précision de ce calcul. Par ailleurs, les cartes d'environnement ne permettent de gérer que les sources lumineuses distantes, donc ne modélisent pas les effets locaux, c'est pourquoi une approche par lancer de photons peut être envisagée.
Ainsi, Jozwowski, dans sa thèse ([Joz02]), propose d'effectuer un lancer de photons en temps réel avec un nombre très restreint de photons. Pour cela, il n'envoie que 100 photons environ par image, mais les conserve d'une image sur l'autre pendant un certain temps jusqu'à obtenir

au maximum 2000 photons. Les photons sont envoyés plus particulièrement aux endroits de la scène ayant été modifiés. Le rendu se fait en temps réel sur un P3 866 MHz pour une fenêtre de 512x512 pixels et un modèle de 256 surfaces. Le calcul de l'illumination est entièrement fait sur le CPU et le rendu en OpenGL. On pourrait espérer obtenir de meilleurs résultats avec un materiel plus récent et des calculs profitant des capacités des nouvelles cartes graphiques, mais il semble difficilement concevable d'obtenir de telles performances sur des modèles de plus de 10 000 éléments de surface même avec le matériel actuel.
Étant donné le peu de photons envoyés, les surfaces spéculaires sont basses et les caustiques sont mal rendues.

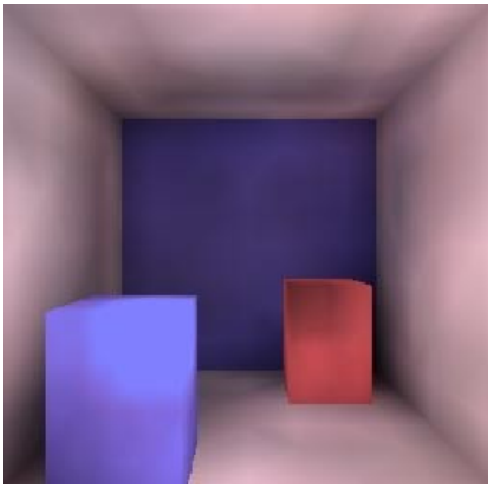

FIG. 2.17 – Boîte de Cornell rendue par lancer de photons en temps réel.

2.9 Radiosité

Les articles précédents concernaient l'éclairage global sur des maillages. Bien qu'ils soient pour la plupart facilement transposables à des modèles basés points, ces articles n'en font pas mention.
[DYN04] nous donne un algorithme pour le calcul de l'éclairage global sur les modèles à base de points grâce au calcul de la radiosité.
Cet algorithme bien que bien mieux adapté aux maillages est donc porté pour les nuages de points en considérant les points comme des disques orientés, et appliquent la méthode d'éléments finis sur ces éléments.
En revanche, l'algorithme de radiosité ne prend en compte que les matériaux diffus. Le résultat est obtenu en quelques minutes de précalcul sur un P4 2.8GHz, et est montré en figure 2.18.

Une approche par radiosité hiérarchique sur des modèles polygonaux a été étudiée par [AH93] pour prendre en compte des matériaux *glossy*. Bien qu'utilisée avec le modèle de BRDF de Cook-Torrance avec un haut degré de spécularité, cette méthode ne marche qu'avec des scènes simples (moins de 5s de calcul pour une subdivision aboutissant à un millier de polygones environ sur un R4000 à 50 MHz). Avec une complexité en théorie de $\mathcal{O}(n \log n)$ et en pratique, selon eux, presque linéaire, elle ne peut être gérable pour les scènes que nous souhaiterions rendre. Il ne

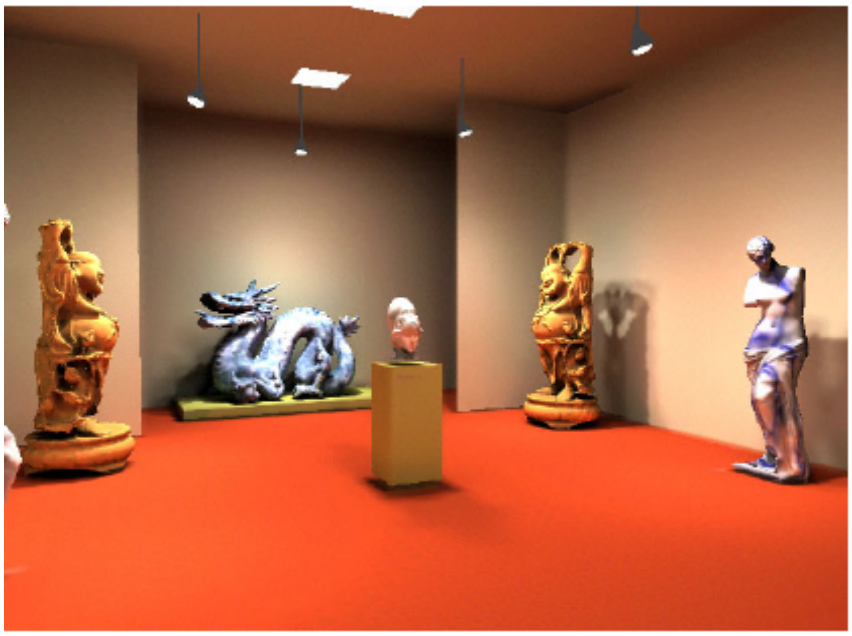

FIG. 2.18 – Galerie rendue avec de la radiosité basée points (351 686 points, 42 min de précalcul - [DYN04]).

s'agit pas en effet d'un précalcul, mais d'un calcul effectué à chaque changement de configuration de la scène (déplacement de la caméra, ou des lumières, ou des géométries...). De plus, la subdivision doit être très fine pour pouvoir prendre en compte les phénomènes hautes fréquences comme les caustiques.
Par ailleurs, cet algorithme de radiosité hiérarchique étant basé sur des subdivisions du maillage, il ne peut être appliqué pour des modèles à base de points. Une version non hiérarchique comme celui de [DYN04] pour un rendu par points et prenant en compte les surfaces spéculaires nécessiterait plutôt un algorithme de complexité de l'ordre de $\mathcal{O}(n^2)$, non applicable sur une scène de plusieurs centaines de milliers de points.

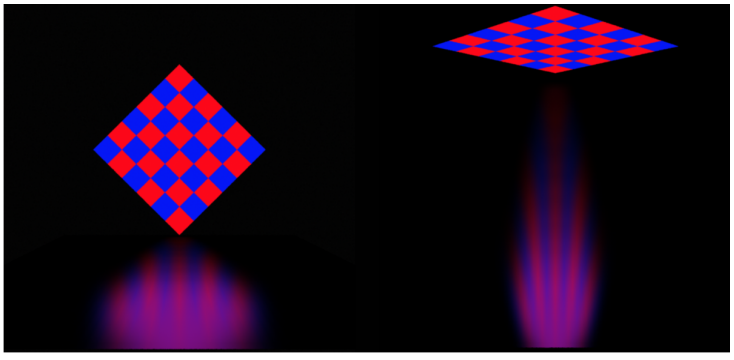

FIG. 2.19 – Scène simple rendue par radiosité hiérarchique prenant en compte les matériaux "glossy".

2.10 Approche hybride Radiosité-Lancer de photons

Granier et al. proposent dans [GDW00] de combiner l'algorithme de radiosité hiérarchique pour précalculer les échanges diffus, et d'envoyer des photons pour pouvoir calculer les caustiques et les effets directionnels, afin de profiter de la rapidité de l'algorithme de radiosité sans avoir le bruit des images calculées entièrement par lancer de photons. Le lancer de photon est accéléré grâce au calcul de la radiosité entre chaque surface ce qui permet de limiter le nombre de photons envoyés.
Le temps de calcul obtenu peut être interactif (3.5 Hz) ou plus long (une dizaine de minutes), suivant la précision souhaitée – sur un R10000 à 200 MHz – pour une scène dont la subdivision aboutit à 80 000 polygones. Pour un temps interactif, les spécularités sont très faibles, mais on commence à deviner la présence de caustiques. Pour des temps de calcul plus longs, les images sont beaucoup plus réalistes mais la scène et le point de vue restent statiques.
De la même manière que pour [AH93], l'approche par radiosité hiérarchique reste difficilement applicable aux modèles à base de points. Un rendu est illustré sur la figure 2.20

Fig. 2.20 – Scène rendue interactivement (3.5 Hz) avec le méthode de [GDW00].

2.11 Résumé

Les performances en temps de calcul, le degré de spécularité des matériaux gérés et les degrés de libertés des algorithmes présentés dans les différents articles étudiés dans l'état de l'art sont résumés dans le tableau 2.1.

Méthode	Sources	Matériaux	Tps de précalcul (indicatif)	Flexibilité
[NRH03]	SD	principalement diffus	3h pour 80k sommets	DC
[SKS02]	SD	basses fréquences	4.4h pour 150k sommets sur un P4 2.2 GHz	DC , RO
[Gre03]	SQ	principalement diffus	NC	DC
[TS06]	SD	glossy	31h49min pour 61k sommets sur un Athlon64 FX-55	DC
[BAOR06]	SD	très spéculaires	15.3 min sur un Intel Xeon64 3.2 GHz avec 8 Go de RAM	EB uniquement
[SLS05]	SD	faiblement glossy	moins de 30min sur un P4 3 GHz	DNR
[KAMJ05]	SP	"très" glossy	3h30 pour 3k triangles sur un cluster de 32 PC 2.8 GHz	DC, DS
[IVD+04]	SD	spéculaires	temps réel (mais éclairage juste plausible)	AC
[GKMD06]	SD	spéculaires	de 1h à 3h sur un cluster de 20 PC	DC
[KM00]	SD	glossy	NC	DC
[Joz02]	SQ	faiblement glossy	temps-réel sur un P3 866 MHz pour 256 surfaces (résultat très approximatif)	AC
[DYN04]	SQ	diffus	42min pour 352k points sur un P4 2.8 GHz	DC
[AH93]	SQ	glossy	5 sec pour 1k polygones sur un R4000 à 50MHz	DC
[GDW00]	SQ	glossy	de 3.5Hz à 10min suivant la précision, pour 80k polygones sur un R10000 à 200 MHz	DC
Notre solution	SQ	spéculaire	moins de 10 min pour 500k points sur un bi-AMD Athlon Dual-Core à 2GHz	DC

Tab. 2.1 – Comparatif des algorithmes - Matériaux : SD : sources distantes (carte d'environnement), SP : sources ponctuelles, SQ : sources quelconques - Flexibilité : DC : Déplacement de la caméra, DS : déplacement des sources lumineuses, RO : Rotation des objets, DNR : Déformation non rigide des objets, AC : aucune contrainte, EB : Édition de la BRDF des objets

3

Solution proposée

3.1 Principe Général

Une première étape de précalcul permet de stocker l'illumination de la scène calculée par lancer de photons : Nous stockons ainsi pour chaque photon sa position, sa couleur et sa direction.
Nous compressons ensuite ces données en représentant la distribution incidente de ces photons par une somme de lobes gaussiens pour chaque splat de la scène. En restreignant la BRDF à celle de Ward isotrope, on utilise ensuite une approximation de l'intégrale de l'équation du rendu en supposant son résultat comme la convolution de deux pseudo-gaussiennes, que l'on supposera être une pseudo-gaussienne en fonction du point de vue de l'utilisateur.
On obtient ainsi un rendu temps-réel si l'on considère la scène statique (sources lumineuses fixes, les matériaux ne peuvent être changés ni les objets déplacés) et un observateur mobile.

3.2 Lancer de photons sur des nuages de points

Les techniques d'éclairage global nécessitent un précalcul de l'éclairage de la scène. Nous avons vu précédemment que ce calcul peut être opéré par radiosité, par l'utilisation d'une carte d'environnement ou par un lancer de photons. Nous avons choisi le lancer de photons afin de pouvoir conserver tous les effets locaux non modélisés par les cartes d'environnement (qui ne considèrent que des sources distantes) et pour garder un temps de calcul raisonnable quel que soit la complexité de la scène, contrairement à la radiosité qui ne permet pas de gérer des scènes trop complexes.

Le lancer de photon sur un nuage de points utilise l'algorithme de raytracing de [SJ00] : il s'agit de calculer l'intersection d'un cylindre représentant le rayon lumineux avec un disque orienté représentant le point (ou "splat"), en pondérant les attributs des intersections trouvées avec la distance radiale au centre du cylindre. Comme l'a vu [AA03], cette méthode ne constitue pas la définition d'une surface puisque la surface varie en fonction de l'incidence du rayon lumineux.
Nous conserverons quand même cet algorithme puisque, bien que cela pouvait poser un problème pour [SJ00] qui l'appliquait à un moteur de lancer de rayon, nous ne l'appliquons ici qu'à un moteur de lancer de photons afin de répartir des photons aléatoirement dans la scène. Nous n'avons donc besoin que d'une faible précision, qui ne justifie donc pas l'utilisation des surfaces

MLS (Moving Least Square) proposée par [AA03] pour décrire précisément la surface définie par les points.

Nous répartissons donc plusieurs millions de photons (de 1 million à 10 millions suivant la qualité que nous souhaitons obtenir) dans la scène en utilisant l'algorithme décrit précédemment, et stockons leur énergie spectrale et leur direction d'incidence. En pratique nous ne stockons leur énergie que selon leur composantes rouge, verte et bleue : cela est insuffisant pour des scènes nécessitant un rendu hautement physiquement réaliste, mais cela reste suffisant pour des scènes visuellement acceptables.

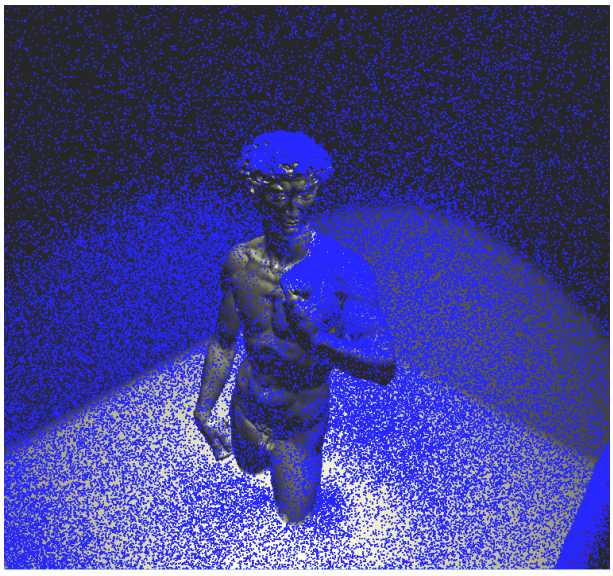

FIG. 3.1 – 200 000 photons (pour plus de lisibilité) répartis dans la scène, affichés directement.

Afin d'accélérer le calcul, les points (ou *splats*) composant les objets de la scène sont disposés dans une grille régulière : le lancer de photon correspond donc à la traversée de cette grille pour chaque photon, et seule l'intersection avec un petit nombre de ces splats (ceux inclus dans les *voxels* traversés) est testée.
Les photons, une fois leur position connue, sont stockés dans une structure de donnée similaire qui facilitera leur accès par la suite.

3.3 BRDF

Pour des raisons que nous évoquerons par la suite, nous nous sommes restreint à des BRDF de Ward (isotropes). La formulation originale de cette BRDF que nous avons vu précédemment et décrite dans [War92] est :

$$f(\theta_i, \varphi_i, \theta_r, \phi_r) = \frac{\rho_d}{\pi} + \rho_s \frac{1}{\sqrt{\cos\theta_i \cos\theta_r}} . \frac{\exp[-tan^2\delta/\alpha^2]}{4\pi\alpha^2}$$

où $\theta_i, \varphi_i, \theta_r, \phi_r$ représentent respectivement $\vec{\omega_i}$ et $\vec{\omega_r}$ en coordonnées sphériques où l'axe z est la normale à la surface, et δ est l'angle entre le *half-vector* et la normale.

Le terme de normalisation $\frac{1}{4\pi\alpha^2}$ est correct pour des α inférieurs à 0.2 (tant que le matériau reste spéculaire).

Nous avions dans un premier temps approximé la paramétrisation avec le *half-vector* en considérant que l'angle entre le *half-vector* et la normale valait la moitié de l'angle entre la direction $\vec{\omega}_r$ et la direction réfléchie de $\vec{\omega}_i$ par rapport à la normale. Cette formule est une formule exacte dans le cas 2D, mais devient une approximation dans le cas 3D. Nous avons finalement laissé tomber cette approche qui donne une erreur importante dans le rendu lorsque les vecteurs $\vec{\omega}_i$, $\vec{\omega}_r$ et la normale ne sont pas dans le même plan.

Par ailleurs, suite à un échange de mails avec Gregory Ward, nous avons pu prendre connaissance d'un article proposant un meilleur facteur de normalisation tel que $\int f(\vec{\omega}_i, \vec{\omega}_r) d\omega_r \approx 1$, et tel que l'échantillonnage stochastique proposé par G.Ward soit plus proche de cette fonction. Il s'agit de l'article non publié de [DÖ5] qui propose la nouvelle BRDF :

$$f(\theta_i, \varphi_i, \theta_r, \phi_r) = \frac{\rho_d}{\pi} + \rho_s \frac{1}{\cos\theta_i \cos\theta_r} . \frac{\exp[-tan^2\delta/\alpha^2]}{4\pi\alpha^2}$$

En effet, lors du lancer de photons, les photons ne sont pas réfléchis avec une probabilité uniforme lorsqu'ils intersectent une surface. Une réflection uniforme nécessiterait un nombre bien plus important de photons afin d'éliminer le bruit. Une méthode pour diminuer la variance est donc de tirer la direction de réflection du photon aléatoirement avec une densité de probabilité (*pdf*) la plus proche possible de la BRDF, et de pondérer l'énergie du photon par $\frac{brdf}{pdf}$.

Ainsi, [War92] proposait un échantillonnage stochastique proche de la partie spéculaire de sa BRDF en tirant aléatoirement la direction réfléchie telle que :

$$\delta = tan^{-1}(\alpha\sqrt{-\log(u_1)})$$

$$\phi = 2\pi u_2$$

avec u_1 et u_2 des variables aléatoires uniformes dans l'intervalle $]0, 1]$, et δ l'angle entre la normale au splat et le *half-vector* et ϕ l'angle de rotation autour de la normale.

Nous avons gardé cet échantillonnage stochastique car [DÖ5] a remarqué qu'il approximait encore mieux la distribution donnée par la BRDF modifiée que la BRDF de Ward initiale. Cela permet donc de réduire encore la variance et ainsi reconstruire une scène moins bruitée.

Par ailleurs, la *pdf* de cette distribution est donnée dans [DÖ5] par :

$$pdf_{[Duer]}(\theta_i, \phi_i, \theta_r, \phi_r) = \frac{\exp[-tan^2\delta/\alpha^2]}{4\pi\alpha^2 \cos\theta_i} . 4 \frac{(1 + \cos\theta_r \cos\theta_i + \sin\theta_r \sin\theta_i \cos(\phi_r - \phi_i)}{(\cos\theta_i + \cos\theta_r)^3}$$

mais nous avons trouvé un rapport technique ([Wal05]) proposant une *pdf* pour l'échantillonnage stochastique précédent donnant la formule :

$$pdf_{[Walter]}(\vec{\omega}_i, \vec{\omega}_r) = \frac{\exp[-tan^2\delta/\alpha^2]}{4\pi\alpha^2(\vec{H}.\vec{\omega}_i)\cos^3\delta}$$

avec $\vec{H}$ le *half-vector*.

Nous avons pu montrer (voir démonstration en annexe) que :

$$pdf_{[Duer]} = pdf_{[Walter]}.\cos\theta_i$$

Ayant soumis ce problème aux deux auteurs Bruce Walter et Arne Duer et n'ayant pas eu de réponse satisfaisantes de leur part, nous avons gardé la version de Bruce Walter qui semble être plus fréquemment utilisée dans la littérature.

De plus, sans perte de généralité, nous avons abusivement considéré la BRDF de Ward comme étant directement multipliée par le terme $\cos\theta_i$ qui apparaît dans la formulation de l'équation du rendu (Eq. 1.2) pour prendre en compte l'angle solide sous lequel on voit la surface éclairée, afin de simplifier les calculs. La nouvelle "BRDF" que nous utilisons donc est :

$$f(\theta_i, \varphi_i, \theta_r, \phi_r) = \frac{\rho_d.\cos\theta_i}{\pi} + \rho_s\frac{1}{\cos\theta_r}.\frac{\exp[-tan^2\delta/\alpha^2]}{4\pi\alpha^2}$$

3.4 Modélisation

Le rendu classique d'une scène dont l'éclairement a été précalculé par lancer de photons nécessite de considérer tous les photons dans un voisinage du point dont on veut faire le rendu, et d'additionner ainsi leurs contributions. Et ce, pour tous les points visibles à l'écran.
En pratique, cette approche ne permet pas d'obtenir un rendu temps-réel, ni même interactif. Cela vient du fait qu'il faut considérer la contribution de chacun des centaines ou milliers de photons pour chacun des points à éclairer. Nous cherchons donc à ne stocker qu'une représentation compacte de cette distribution de photons incidents.
Nous avons vu que les approches précédentes qui consistaient pour la plupart à projeter cette distribution dans un base (harmonique, d'ondelettes, etc...) ne pouvait prendre en compte que des distributions basses fréquences en ne gardant qu'un nombre raisonnable de coefficients. Nous avons donc choisi d'utiliser une représentation compacte de la distribution de photons pouvant prendre en compte de hautes fréquences : une distribution sous forme de somme de lobes gaussiens. Par ailleurs, on considère un éclairage constant par splat ce qui est raisonnable pour des tailles de splats "petits" par rapport à la taille de la scène (le rendu par splat n'est lui même adéquat que dans l'hypothèse de splats "petits").

Nous écrivons donc l'expression de la densité d'un lobe :

$$g(\vec{\omega}_r) = \frac{\exp(-\frac{tan^2\delta}{\beta^2})}{4\pi\beta^2}$$

avec β l'écart-type du lobe et δ l'angle entre le "half-vector" $\vec{H}$ et la normale $\vec{N}$ au splat. On définit ainsi un "half-vector" comme le vecteur médiateur entre le vecteur $\vec{\omega}_r$ et la direction réfléchie $\vec{L}'$ du lobe $\vec{L}$.

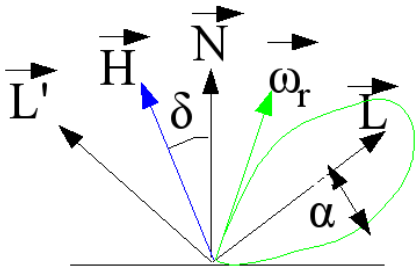

FIG. 3.2 – Lobe autour de la direction $\vec{L}$

En pratique, étant donné les restrictions du nombre de variables que l'on peut envoyer à la carte graphique, et les contraintes de temps de calcul, nous avons choisi d'utiliser une somme de 5 lobes gaussiens pour représenter la distribution de photons incidents. On exprimera donc la distribution incidente sous la forme suivante :

$$\mathcal{N}(\vec{\omega}_r) = \sum_{i=1}^{5} \lambda_i \frac{\exp(-\frac{tan^2\delta_i}{\beta_i^2})}{4\pi\beta_i^2}$$

et on notera :

$$\mathcal{N}(\vec{\omega}_r) = \sum_{i=1}^{5} \lambda_i \mathcal{N}_i(\vec{\omega}_r, \vec{L}_i, \beta_i)$$

avec $\vec{L}_i$ la direction du lobe i.

3.5 Algorithme Expectation-Maximization

Étant donnée une distribution de photons incidents dans un disque de rayon donné, il s'agit d'estimer au mieux les paramètres des 5 lobes : λ_i, $\vec{L}_i$ (= 2 paramètres, puisque c'est une direction) et β_i. Cela fait donc 5 lobes à 4 paramètres chacun à optimiser, soient 20 paramètres.

Il existe de nombreuses méthodes d'optimisation non linéaires pour résoudre un tel problème : des méthodes déterministes (Levenberg-Marquardt, descentes de gradient...), et stochastiques (Expectation-Maximization et variantes...).
Les méthodes déterministes peuvent être lentes à évaluer pour des recherches à hautes dimensions (20 dimensions ici) et sont lourdes à mettre en oeuvre. En effet, une scène pouvant comporter 1 million de splats (et dans ce cas il y aura 1 million d'optimisations à effectuer), et le périmètre de recherche des photons pouvant contenir jusque 1 à 5 000 photons dont on essaiera de trouver la distribution, cette optimisation doit être très rapide à calculer.

Dans un premier temps nous avons essayé un algorithme de classification : l'algorithme de quantification vectorielle (ou k-moyennes) pour trouver 5 classes de photons. Cette approche n'a pas été probante. Ce peut être dû aux mélanges de gaussiennes, mal géré par une telle méthode qui permet de séparer des classes si elles sont effectivement distinctes.

Nous avons finalement opté pour l'algorithme d'Expectation-Maximization (EM) qui fonctionne bien dans le cas de mélanges gaussiens et qui consiste à maximiser la *log-vraisemblance* de l'échantillon (nous verrons son expression par la suite). Nous utiliserons l'implémentation donnée par [Cro].
Cet algorithme introduit une variable cachée qui est la probabilité pour chaque individu (ici chaque photon) d'appartenir à l'une des 5 gaussiennes. Nous appellerons cette probabilité $h(m,n)$, probabilité que le photon m appartienne à la gaussienne n. On notera M le nombre de photons dans le voisinage considéré.

Partant d'un état initial (estimation des paramètres, ou paramètres aléatoires), l'algorithme itère les étapes :

- Estimation de la variable cachée $h(m,n)$:

$$h(m,n) = \frac{\lambda_n \mathcal{N}_n(\vec{\omega}_m, \vec{L_n}, \beta_n)}{\mathcal{N}(\vec{\omega}_m)}$$

- Maximisation de la log-vraisemblance pour obtenir les autre variables en utilisant $h(m,n)$ calculé précédemment :

$$\lambda_n = \frac{1}{M} \sum_{m=1}^{M} h(m,n)$$

$$\beta_n = \frac{\sum_{m=1}^{M} h(m,n) tan^2 \delta_m}{\sum_{m=1}^{M} h(m,n)}$$

$$\vec{L_n} \approx \frac{\sum_{m=1}^{M} h(m,n) \vec{\omega_m}}{\sum_{m=1}^{M} h(m,n)}$$

Les estimations de λ_n et β_n maximisent réellement la log-vraisemblance, alors que l'estimation de $\vec{L_n}$ n'est pas exacte : une estimation exacte nécessiterait la résolution d'un système d'équations non linéaires. En pratique, nous nous contentons d'estimer ce paramètre (la direction du lobe gaussien n) comme une "moyenne pondérée des directions incidentes", avec une pondération directement proportionnelle à la contribution du photon à ce lobe, ce qui est intuitivement plausible.

De plus, il n'est pas idéal de considérer que la contribution des photons aux lobes est la même pour tous les photons : en effet, beaucoup de photons ont une énergie très faible et ne doivent pas contribuer autant que les photons de forte énergie.
Nous utilisons donc un poids $w(i)$ associé à chaque photons, qui est simplement la moyenne de ses composantes rouges, vertes et bleues, et nous avons remarqué que le prise en compte de ces poids revenait simplement à pondérer le tableau $h(m,n)$ par les poids $w(m)$:

$$h(m,n) \leftarrow h(m,n) w(m)$$

et à modifier l'estimation de λ_n par :

$$\lambda_n = \frac{\sum_{m=1}^{M} h(m,n)}{\sum_{m=1}^{M} w(m)}$$

(en considérant les $h(m,n)$ déjà mis à jours par la pondération).

Par ailleurs nous cherchons à obtenir une reconstruction colorée. Ainsi, au lieu d'introduire un coefficient λ_i par couleur ce qui augmenterait la dimension de l'espace de recherche à 30 dimensions au lieu de 20 (on aurait 3 coefficients pour pondérer chaque lobe, suivant leur composante rouge, verte et bleue au lieu d'un seul), on estime la couleur de chaque lobe à posteriori. La couleur de chaque lobe ne faisant pas partie de l'optimisation, l'estimation à posteriori de la couleur du lobe sera plus enclin à nous donner des mélanges indésirables de couleurs (si deux lobes de couleurs différentes ont la même direction et le même écart-type, ils seront fusionnés en un seul lobe de couleur moyenne, alors qu'en intégrant la couleur dans l'optimisation, nous aurions eu deux lobes distincts... mais le résultat sera le même lors de l'évaluation !). On peut aussi penser à juste titre que d'une direction donnée provient majoritairement des photons de couleur similaire dans la plupart des cas.

Ainsi, à chaque lobe est associée une couleur :

$$c_i = \frac{\sum_{m=1}^{M} h(m,n) c_m}{\sum_{m=1}^{M} h(m,n)}$$

avec c_m la couleur (ou énergie spectrale) de chaque photon.

Un problème essentiel de l'algorithme EM est sa convergence vers un maximum local de la log-vraisemblance. Nous avons alors essayé plusieurs variantes de l'algorithme EM afin d'accélérer l'algorithme et/ou de trouver un maximum global, parmis un vaste choix de variantes existantes (AEM, SEM, SAEM, SpEM, IEM, SpIEM, Lazy EM, OSEM...) :

- Recuit simulé sur les paramètres initiaux
  Une première approche consiste à faire converger l'algorithme EM à partir de plusieurs conditions initiales et de ne retenir que les valeurs améliorant la log-vraisemblance. Nous avons opté pour une approche légèrement différente faisant intervenir l'aspect de recuit simulé :
  On itère plusieurs fois l'algorithme EM complet. On utilise le résultat de l'itération précédente que l'on bruite comme état initial de l'itération courante. L'intensité du bruit diminue à chaque itération afin de simuler un recuit. A chaque itération est estimée la log-vraisemblance par :

  $$L = \sum_{i=1}^{M} \log \mathcal{N}(\vec{\omega}_i)$$

  Nous conservons le résultat si et seulement si la log-vraisemblance a été améliorée par rapport à la solution précédemment retenue. Il s'agit d'une solution intuitive et sa convergence n'a pas été démontrée.

- SAEM
  Ou *Simulated Annealing Expectation Maximization*, proposé pour la première fois par [CD91]. Elle consiste à intercaler entre les étapes "Expectation" et "Maximization" deux nouvelles étapes : Une étape de simulation d'une loi multinomiale d'ordre 1 pour chaque individu et de paramètres $h(m,n)$, et une étape consistant à "bruiter" le tableau $h(m,n)$ en fonction de la variable aléatoire de loi multinomiale simulée et d'une température qui

décroît. En revanche, nous n'avons pas supprimé les lobes de contributions trop faibles comme proposé par [CD91]. Par ailleurs, [CD91] conseille une température qui décroît en $\cos(itration/Nombre_itrations)$, qui est une décroissance plus lente qu'une décroissance linéaire testée aussi.
On notera qu'une loi multinomiale d'ordre 1 est à valeur dans {0, 1} et sera donc ici un vecteur de 5 composantes, toutes nulles sauf une.

On obtient donc l'algorithme :
- "Expectation" classique
- $\gamma_i = \cos(\frac{\pi}{2}\frac{i}{I})$
- Pour chaque individu m (= photon) :
k = indice de la valeur non nulle d'une variable aléatoire de loi multinomiale d'ordre 1 et de paramètre $h(m,:)$.
Pour chaque lobe n :
Si n==k alors
$h(m,n) = h(m,n) + \gamma_i * (1 - h(m,n))$
Sinon
$h(m,n) = h(m,n) - \gamma_i * h(m,n)$
Fin Si
Fin Pour
Fin Pour
- "Maximization" classique

Cet algorithme permet d'assurer un optimum global de la log-vraisemblance si la décroissance de la température est assez faible.

– SpEM
Ou *Sparse Expectation Maximization* proposé pour la première fois par [NH98] . Cet algorithme vise à réduire les temps de calcul liés à l'Expectation qui peut être coûteuse.
Il s'agit simplement de ne pas mettre à jour les valeurs du tableau $h(m,n)$ pour les photons dont la contribution à un lobe est trop faible, et donc de conserver son ancienne valeur sur plusieurs itérations.
Nous avons conservé cette approche en ne forçant la mise à jour de tout le tableau $h(m,n)$ toutes les 8 itérations, et à ne pas recalculer $h(m,n)$ pour les photons m dont la contribution au lobe n est telle que $h(m,n) < 0.02$. Nous avons utilisé l'implémentation proposée par [NM03].

En revanche nous n'avons pu tester d'autres variantes :

– IEM (Iterative EM), OSEM (Ordered Subset EM), SpIEM (Sparse Iterative EM), approches par kd-tree...
Ils consistent à ne mettre à jour qu'une partie des données par itération. Dans notre cas, nous n'utilisons qu'un nombre fixe et faible d'itérations (de 5 à 20 itérations seulement) ce qui ne permettrait pas de parcourir tous les photons correctement si pour chaque itération une partie seulement était explorée.

– CEM (Classification EM)

Comme l'algorithme de quantification vectorielle que nous avons testé, il s'agit d'un algorithme porté vers la classification des individus et ne correspond donc pas à l'approche par somme de gaussiennes que nous nous sommes fixée.

– SEM (Stochastic EM)
Nous avons pu tester l'algorithme SAEM qui est une amélioration de l'algorithme SEM proposée par les mêmes auteurs ([CD91]). L'algorithme SEM propose juste d'ajouter un individu tiré aléatoirement avec une loi multinomiale à la liste des individus et d'opérer à l'étape de maximisation en utilisant ce nouvel individu.

Obtenant un bruit plus important par les méthodes stochastiques (SAEM et recuit simulé sur les conditions initiales) mais la qualité visuelle n'étant que peu améliorée, nous avons opté pour l'algorithme SpEM sans étape stochastique.
Nous avons gardé la version "Sparse" car elle améliorait le temps de calcul pour un résultat imperceptiblement différent.

Par ailleurs, nous avons testé plusieurs manières d'obtenir des estimations initiales :
– Paramètres aléatoires
En choisissant des paramètres entièrement aléatoire à chaque splat, l'algorithme EM converge vers des solutions différentes d'un splat à l'autre. Nous obtenons donc un rendu bruité.

– Utilisation d'une *cubemap*
Nous discrétisons l'espace 2D des directions des lobes en facettes d'un cube unitaire de 2x2x2 facettes (chaque dimension est divisée en deux). Ainsi, pour chacune des 24 facettes de ce cube (chacune des 6 faces du cube étant divisée en 4 facettes), nous pouvons comptabiliser le nombre de photons se dirigeant vers elle.

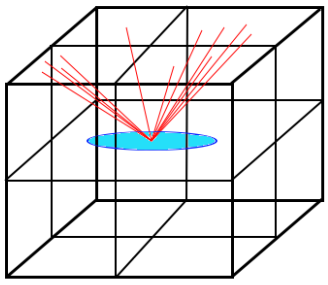

FIG. 3.3 – Photons (en rouge) intersectant un splat (en cyan). Leurs directions sont stockées dans les faces de la cubemap (en noir).

On trie alors les 24 facettes du cube et nous pouvons alors garder comme directions initiales de l'algorithme EM le centre des 5 facettes comportant le plus de photons, et comme

contribution de chaque lobe le nombre de photons intersectant ces 5 faces sur le nombre de photons dans la cubemap.
Une approche similaire consiste à distribuer les photons en fonction de leur distance (nous avons choisi une distance orthodromique) aux sommets du cube, et trier non plus les faces du cube mais leurs sommets.
Dans tous les cas, cette méthode n'a pas donné de résultats convaincants : Si un lobe est à cheval sur 2, 3 ou même 4 faces (dans le cas d'un lobe autour de la normale au splat par exemple), les directions principales trouvées seront très proches et l'algorithme EM les fusionnera. Nous avons le même type de problème sur l'approche par sommets et non par faces.

- Réutilisation du résultat du splat précédent
  Sous la condition que les splats soient ordonnés spatialement (deux splats consécutifs dans la liste des splats sont voisins spatialement), on peut réutiliser le résultat de l'algorithme EM du splat précédent comme état initial du splat courant. Cela permettrait d'améliorer le résultat (et donc diminuer le nombre d'itérations nécessaire à la convergence de l'algorithme EM) et d'améliorer la continuité des lobes entre les splats.
  Ayant testé cette approche, elle n'a pas non plus donné de résultats satisfaisants car lorsqu'une fusion ou une suppression de lobe a lieue sur un splat (à tort ou à raison), nous obtenons des directions initiales identiques sur le splat suivant, de telle manière qu'il ne restera plus que quatre lobes sur ce splat... et ainsi de suite jusqu'à ce qu'il ne reste plus qu'un seul lobe par splat... et c'est effectivement ce que nous obtenons, à savoir un seul à deux lobes par splat pour la majorité des splats au lieu de cinq.

- Paramètres constants
  Les paramètres (directions, écart-type...) sont choisis arbitrairement de manière à couvrir uniformément le demi-espace des directions. Au changement de repère près, chaque splat possède donc les mêmes conditions initiales, ce qui permet de faire converger l'algorithme EM vers des états voisins pour des splats partageant des zones de recherches de photons voisines.
  Nous avons finalement opté pour cette méthode pour la qualité visuelle des images données par rapport aux trois méthodes précédentes.

Les choix SpEM + conditions initiales constantes nous permettent d'obtenir un résultat visuellement acceptable en généralement moins de 10 itérations.
Nous avons ainsi choisi un nombre constant d'itérations ce qui nous évite un calcul très coûteux de la log-vraisemblance à chaque itération. Ce choix est justifié dans le chapitre "résultats".
A ce point nous connaissons donc les paramètres (direction, écart-type, et couleur) de chacun des 5 lobes en chaque splat de la scène.

3.6 Méthode de rendu

3.6.1 Évaluation de l'équation du rendu

Nous connaissons la distribution approchée $\mathcal{N}(\vec{\omega}_i)$ de photons entrant sur chaque splat ainsi que la BRDF $f(\vec{\omega}_i, \vec{\omega}_r)$ du matériau du splat.

L'équation du rendu s'exprime alors par :

$$L(\vec{\omega_r}) = \int_{\Omega_i} \mathcal{N}(\vec{\omega}_i) f(\vec{\omega}_i, \vec{\omega}_r) d\vec{\omega}_i$$

et doit être évaluée très rapidement pour pouvoir afficher la scène en temps réel.

Cette équation s'écrit donc de manière complète (en omettant la partie diffuse) :

$$L(\vec{\omega_r}) = \int_{\Omega_i} \left[\left(\sum_{k=1}^{5} c_k . \frac{e^{-\frac{\tan^2 \delta_k}{\beta_k^2}}}{4\pi\beta_k^2} \right) . \frac{e^{-\frac{\tan^2 \delta}{\alpha^2}}}{4\pi\alpha^2 \cos\theta_r \cos\theta_i} \cos\theta_i \right] d\vec{\omega}_i$$

avec δ_k l'angle entre le "half-vector" de chaque lobe k (que l'on a défini comme le vecteur médian entre la direction réfléchie du lobe k par rapport à la normale et la direction $\vec{\omega}_i$) et la normale du splat, et δ l'angle entre le half-vector (vecteur médian entre le $\vec{\omega}_i$ et $\vec{\omega}_r$) et la normale du splat. c_k représente la couleur du lobe k.

Même en sortant le signe somme de l'intégrale, on voit que cette intégrale n'admet pas de solution analytique évidente. De plus, une évaluation numérique (par intégration de Monte-Carlo par exemple) serait trop coûteuse pour être effectuée en temps réel sur chacun des dizaines milliers de splats composant la scène.

En nous appuyant sur les résultats de [DHS+05], nous utilisons le fait que dans le cas de BRDF qui s'expriment par une différence entre la direction spéculaire pure (direction réfléchie de $\vec{\omega}_i$ par rapport à la normale au splat) et $\vec{\omega}_r$, l'intégrale souhaitée est la convolution entre la distribution incidente et la BRDF pour $\vec{\omega}_r$ fixé.
La BRDF de Ward modifiée et incluant le terme d'angle solide $\cos\theta_i$ répond approximativement à ce critère (nous avons vu que dans le cas 2D, l'angle entre le "half-vector" et la normale est justement la moitié de l'angle entre la direction spéculaire pure et $\vec{\omega}_r$, mais il n'y a pas de paramétrisation explicite simple dans le cas 3D).

On approximera donc l'équation du rendu par :

$$L(\vec{\omega_r}) = \sum_{k=1}^{5} c_k \left(\mathcal{N}_k(\vec{\omega}_r, \vec{L_k}, \beta_k) * f(\vec{\omega}_i, \vec{\omega}_r) \right)$$

Par ailleurs, le choix de lobes pseudo-gaussiens pour représenter la distribution incidente de photons et d'une BRDF pseudo-gaussienne (à $\vec{\omega}_r$ fixé) nous laisse penser que, comme d'une manière générale une gaussienne de variance σ_a^2 convolée avec une gaussienne de variance σ_b^2 est une gaussienne de variance $\sigma_a^2 + \sigma_b^2$, nous pourrons approximer correctement l'équation du rendu par un lobe gaussien.

Nous avons considéré que cette approximation était bonne pour la plupart des valeurs de β_k et de $\vec{N}.\vec{L_k}$ (cf. résultats) mais cette approximation n'est correcte qu'à un facteur multiplicatif près, estimé approximativement à 1.8.

Nous obtenons donc finalement l'expression de l'éclairage dans la direction $\vec{\omega}_r$ par :

$$L(\vec{\omega_r}) = \sum_{k=1}^{5} c_k . \frac{e^{-\frac{\tan^2 \delta}{\alpha^2+\beta_k^2}}}{4\pi(\alpha^2+\beta_k^2)}$$

ce qui peut être évalué par la carte graphique plusieurs millions de fois par seconde, donc suffisamment rapidement pour afficher des dizaines de milliers de splats à plusieurs images secondes (voir résultats).

3.6.2 Tone Mapping

Le résultat de ces précédentes opération nous donne une luminance dans une direction donnée. Cette luminance est une grandeur physique (une intensité lumineuse, en candela) et la perception que l'on en a n'est pas directement proportionnelle à cette grandeur mais logarithmique.
Par ailleurs, l'oeil s'adapte en contractant la pupille lorsque qu'une scène naturelle est très lumineuse, ou en la dilatant lorsqu'il fait sombre, et, lorsqu'il fait sombre les récepteurs de la rétine les plus solicités sont les bâtonnets qui n'ont qu'une vision monochromatique (vision photoptique). De plus, les écrans LCD ou cathodiques ne permettent d'afficher qu'un intervalle d'intensités lumineuses restreint.

Tous ces phénomènes nous amènent à utiliser un opérateur de *Tone Mapping* pour ramener l'intensité lumineuse de l'équation du rendu à un intervalle compris entre 0 et 1, ce que l'écran peut afficher.

Nous avons choisi l'opérateur de [KMS05] pour sa simplicité de mise en oeuvre et pour sa compatibilité avec le rendu temps-réel. Nous avons utilisé une correction "globale" des luminances (qui utilise un facteur dépendant de l'image complète, à l'inverse d'une version locale qui utiliserait un facteur de correction dépendant uniquement du voisinage du pixel à évaluer), en négligeant les aspects dynamiques (adaptation temporelle) et fréquentiels (halos autour de sources brillantes, perte de netteté en vision photoptique).

L'algorithme est le suivant :

- Effectuer un rendu de la scène dans une image de 32 bits par composante rouge, verte et bleue, avec des intensités "non bornées".
- Calculer $\bar{Y} = \exp(\frac{1}{N}\sum_{i=1} N \log(\epsilon + Luminance(pixel_i)))$ avec $Luminance(pixel_i) = 0.33 * pixel_i.rouge + 0.71 * pixel_i.vert + 0.08 * pixel_i.bleu$ et ϵ une valeur faible ($<< 1$) évitant à la somme de diverger en présence de pixels noirs. Ce calcul est effectué par GPGPU (code exécuté directement par la carte graphique) récursivement.

- Pour chaque pixel i (dans un *pixel shader*, géré par la carte graphique)
  - Calculer $Y_r = \frac{Luminance(pixel_i)*\alpha}{\bar{Y}}$ avec la constante $\alpha = 0.18$ (valeur donnée par [RSSF02]).
  - Calculer $L = \frac{Y_r}{1+Y_r}$
  - Calculer $\sigma = \frac{0.04}{0.04+Y}$ la fraction de lumière reçue par les cônes par rapport à celle

reçue par les bâtonnets (ratio vision photoptique/scotopique).

- Retourner la couleur finale : $couleur_f = \frac{L}{Y} * pixel_i * (1-\sigma) + [1.05, 0.97, 1.27]' * L * \sigma$. Le vecteur $[1.05, 0.97, 1.27]'$ représente une couleur légèrement bleuté que l'oeil perçoit en vision nocturne.

Fin Pour

4

Résultats

4.1 Approximation de l'équation du rendu

Nous avons comparé numériquement le calcul de l'intégrale :

$$L(\vec{\omega_r}) = \int_{\Omega_i} \left[\left(\frac{e^{-\frac{\tan^2 \delta_0}{\beta_0^2}}}{4\pi\beta_0^2} \right) . \frac{e^{-\frac{\tan^2 \delta}{\alpha^2}}}{4\pi\alpha^2 \cos\theta_r \cos\theta_i} \cos\theta_i \right] d\vec{\omega_i}$$

qui est l'expression de la luminance donnée par un seul lobe, avec l'approximation :

$$L(\vec{\omega_r}) = \frac{e^{-\frac{\tan^2 \delta}{\alpha^2+\beta_0^2}}}{4\pi(\alpha^2+\beta_0^2)}$$

Pour cela nous avons échantillonné stochastiquement la demi-sphère des directions $\vec{\omega_i}$ et $\vec{\omega_r}$ tels que $\vec{\omega_{ietr}}.\vec{N} > 0$ de manière uniforme, c'est à dire en générant des vecteurs $\vec{\omega_i}$ et $\vec{\omega_r}$ tels que :

$$\vec{\omega_i} = \begin{pmatrix} \cos(2\pi r_1)\sqrt{1-r_2^2} \\ \sin(2\pi r_1)\sqrt{1-r_2^2} \\ r_2 \end{pmatrix}$$

, avec r_1 et r_2 des variables aléatoires uniformes sur $[0, 1]$, et nous avons calculé la première intégrale par une somme discrète sur les directions $\vec{\omega_i}$ et ce, pour plusieurs directions $\vec{\omega_r}$. Une meilleure approche aurait été de générer les vecteurs $\vec{\omega_i}$ avec une densité de probabilité proche de la valeur de l'intégrale afin de diminuer la variance, mais nous avons opté par simplicité pour une génération d'un plus grand nombre d'échantillons (10 000 échantillons pour $\vec{\omega_i}$ et 1 500 pour $\vec{\omega_r}$). Nous avons ainsi effectué cette opération pour α fixé à 0.065, et pour β variant de 0.01 à 0.5 (avec 21 valeurs de β), ainsi que pour une direction de lobe dont l'angle θ avec la normale prenait 19 valeurs de 0 degré à 90 degrés.

Nous avons pu comparer avec l'approximation et tracer l'erreur en norme $\mathcal{L}^2$ ainsi que le facteur d'échelle entre les deux fonctions (nous avons vu en effet que ces deux fonctions étaient proches si on considérait un facteur d'échelle) pour chaque β et chaque angle θ.

Pour obtenir ce ratio, nous avons calculé le quotient des moyennes des fonctions de l'intégrale numérique et de l'approximation par une gaussienne. Nous observons un bruit faible dans

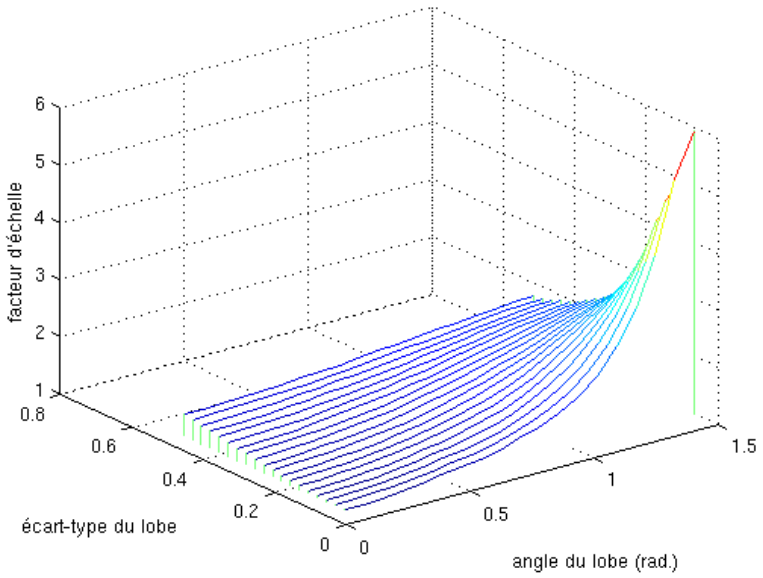

FIG. 4.1 – Facteur d'échelle entre l'approximation numérique de l'intégrale et son approximation par une gaussienne en fonction de β et θ

ce calcul qui provient de l'échantillonnage stochastique qui est opéré avec un générateur pseudo-aléatoire, donc les échantillons sont les mêmes d'une simulation à l'autre.

Nous traçons alors l'erreur relative en norme $\mathcal{L}^2$ entre ces deux fonction, chacune divisée par sa moyenne (par β et par θ). Nous obtenons le résultat de la figure 4.2.

Nous voyons que, malgré les 10 000 échantillons pour évaluer l'intégrale sur la demi-sphère, il subsiste un bruit important.

En considérant le ratio constant (la moyenne de ce ratio pour tous les β et tous les θ vaut 1.81), nous pouvons alors calculer l'erreur relative en norme $\mathcal{L}^2$ avec l'approximation gaussienne divisée par la constante 1.81. Cette considération est abusive, mais une fonction non constante de β et de θ pour ce ratio est en cours d'évaluation. Nous obtenons alors la figure 4.3.
Il s'agit donc de l'erreur relative par lobe que nous commettons en rendant l'image. Nous observons que la plupart des erreurs commises sont entre 5% et 35% ce qui pourra être amélioré en considérant le facteur d'échelle non constant. Nous nous penchons actuellement sur ce problème.

Par ailleurs, le tracé d'un lobe incident, de l'approximation par intégration numérique et de l'approximation par lobe gaussien ne permet pas de détecter cette erreur. En effet, les deux approximations sont très similaires, quels que soient β et θ. Nous pouvons voir ce tracé en figure 4.4, dans le cas où l'erreur est censée être proche des 30%.

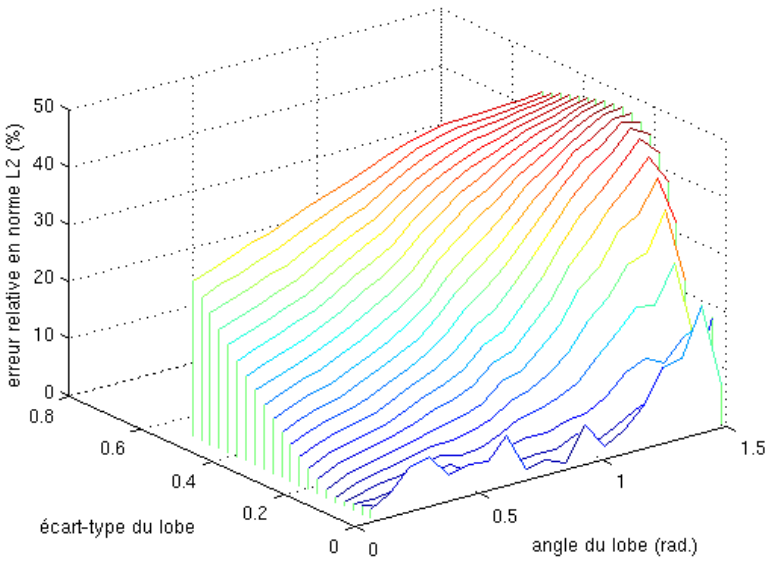

FIG. 4.2 – Erreur relative en norme $\mathcal{L}^2$ (en %) entre l'intégrale du rendu et son approximation gaussienne (en utilisant le ratio exact en chaque point)

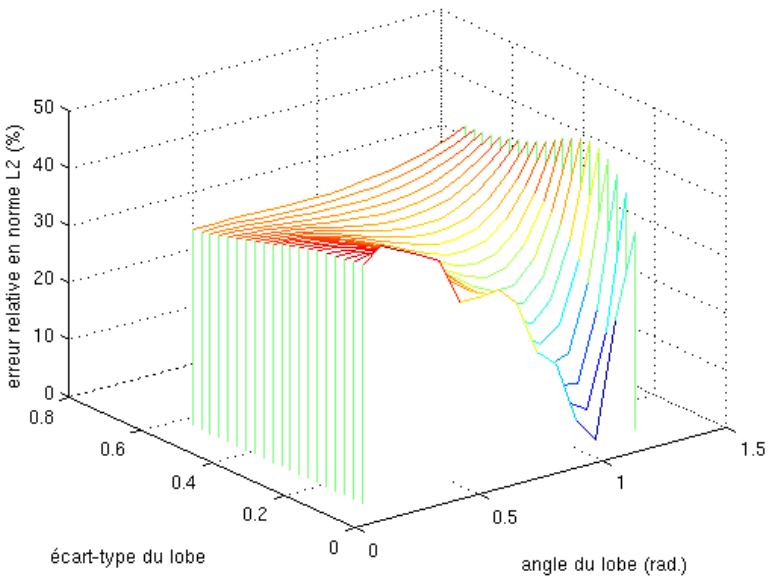

FIG. 4.3 – Erreur relative en norme $\mathcal{L}^2$ (en %) entre l'intégrale du rendu et son approximation gaussienne

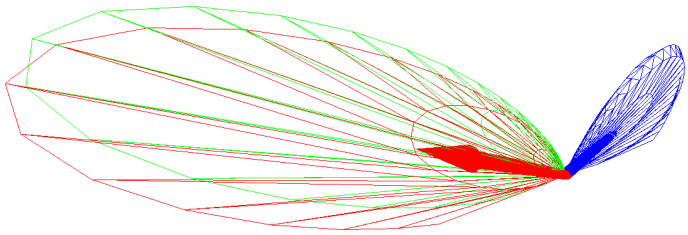

FIG. 4.4 – Tracé du lobe incident (bleu) et des approximations numériques et par lobe (vert et rouge).

4.2 Comparaison avec une référence

Nous pouvons comparer le rendu d'une image de référence calculée par l'évaluation directe de la luminance en chaque point grâce à la carte de photons pour un point de vue fixé avec le rendu temps réel de notre approximation.
Nous obtenons alors pour une scène entièrement spéculaire la figure 4.5

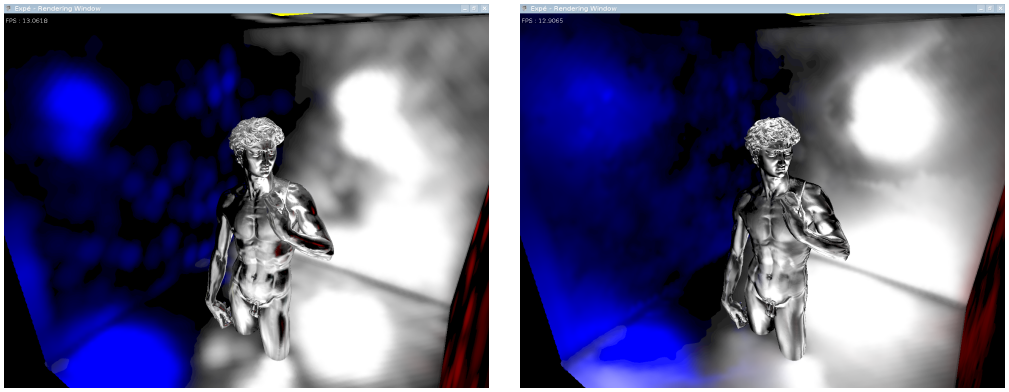

FIG. 4.5 – Comparaison image de référence (gauche)/approximation (droite)

Nous observons que nous perdons de l'information haute fréquence et que nous avons aussi perdu un peu en localisation des reflets.

4.3 Temps de précalcul et résultats

Le code a été parallélisé afin de fonctionner sur un ordinateur possédant 2 processeurs dual-core (équivalent donc à 4 processeurs) en répartissant le précalcul sur 4 threads. En effet, chaque splat est indépendant ce qui rend aisé cette parallélisation.
En pratique, les résultats donnés ont été obtenus grâce à un AMD Opteron Dual Core 270, à 2 GHz.

Nous comparerons les temps de calcul par rapport à une reconstruction de référence obtenue avec 1 million de photons envoyés sur le modèle "Igea" dans une "boite de Cornell" totalisant 67 601 splats, après un lancer de photons de 52 secondes, et une optimisation de 23 secondes de 10 itérations de l'algorithme EM. (4.6)

Le matériau est le même pour les deux objets, diffus à 10%, spéculaire à 70% et dont l'écart-type du materiau de Ward est $\alpha = 0.065$. Nous ne conserverons que l'éclairage indirect. Ainsi, après la phase de lancer de photons, nous avons 698 515 photons (les autres ayant été absorbés entre le premier et le deuxième rebond de ceux-ci). La zone de recherche par splat est de 3 unités (à titre de comparaison, la largeur de la Cornell-Box est de 120 unités). Pour chaque splat, nous obtenons ainsi un nombre moyen de 69 photons par splat.

Les images présentées n'ont pas subi la phase de Tone Mapping afin de comparer plus facilement les résultats.

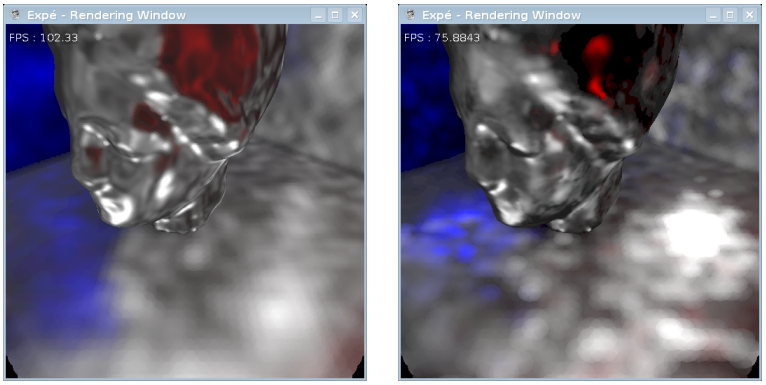

FIG. 4.6 – Image de référence par utilisation de la carte de photons (3 millions de photons) ainsi que la reconstruction servant de référence pour les temps de calcul

Nous avons augmenté la luminosité afin que les images soient suffisamment explicites (l'éclairage indirect affiché seul est beaucoup plus faible, et donc le bruit moins visible).

4.3.1 en fonction du nombre d'itération de l'algorithme EM

Nous avons fait varier le nombre d'itérations de l'algorithme EM. Les résultats obtenus sont en figure 4.7.

Nous observons que dès 5 itérations, le temps de calcul est en $\mathcal{O}(n)$ avec n le nombre d'itérations. La faible différence de temps de calcul entre l'algorithme en 2 itérations et en 5 itérations s'explique par le fait qu'à chaque splat, il faille chercher les photons dans une certaine boule autour du splat, ce qui a un coût constant non nul.
Nous observons aussi que le nombre d'itérations joue sur le spécularité : les hautes fréquences sont mieux prises en compte avec 10 itérations (référence) que 2 itérations, mais nous voyons qu'il n'y a pas de changements significatifs entre 10 itérations et 100 itérations, ce qui justifiera le choix d'un critère d'arrêt fixe.

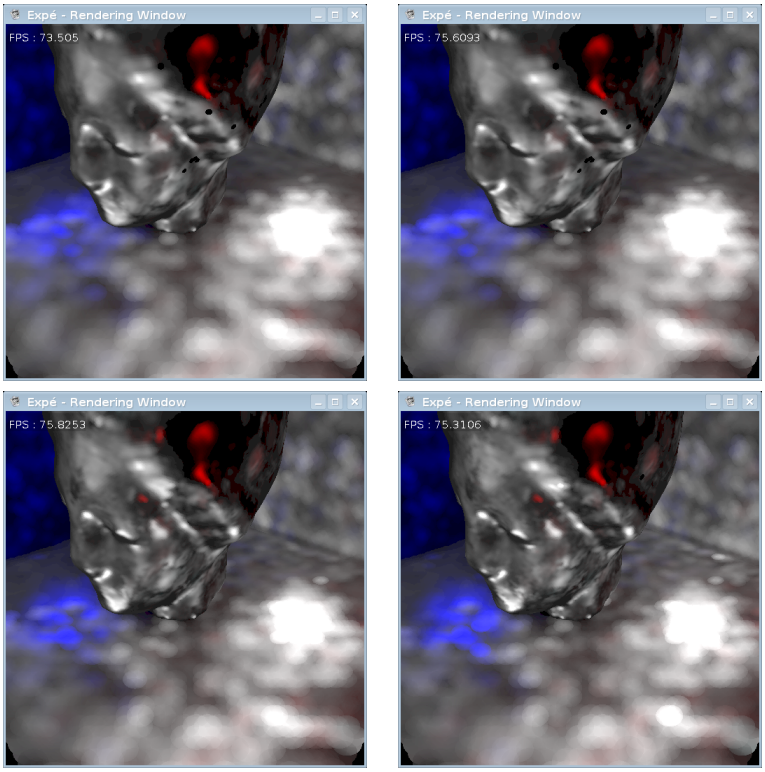

FIG. 4.7 – Comparaison des itérations de l'algorithme EM : 2 itérations (9 secondes pour l'optimisation), 5 itérations (13 secondes), 20 itérations (44 secondes) et 100 itérations (3min30)

4.3.2 en fonction du nombre de photons envoyés

Nous avons fait varier le nombre de photons initialement envoyés. Les résultats obtenus sont en figure 4.8.

Nous voyons que le nombre de photons envoyés joue énormément sur le bruit de l'image. De plus, la complexité pour le lancer de photons ainsi que pour l'optimisation est en $\mathcal{O}(N)$ avec N le nombre de photons envoyés ou reçus.
Un bon compromis (pour cette scène et ces matériaux) se situerait entre 1 million de photons (référence) et 3 millions.

4.3.3 en fonction du nombre de splats

On fait maintenant varier le nombre de splats de la scène. Nous avons utilisé une version du modèle "Igea" discrétisé plus finement (figure 4.9).

On remarque que le lancer de photons s'est fait en temps presque constant, ce qui s'explique par la structure de donnée (une grille volumique) stockant les points de la scène. Les photons ont juste à tester un nombre restreint d'intersections : l'intersection du photon avec les voxels,

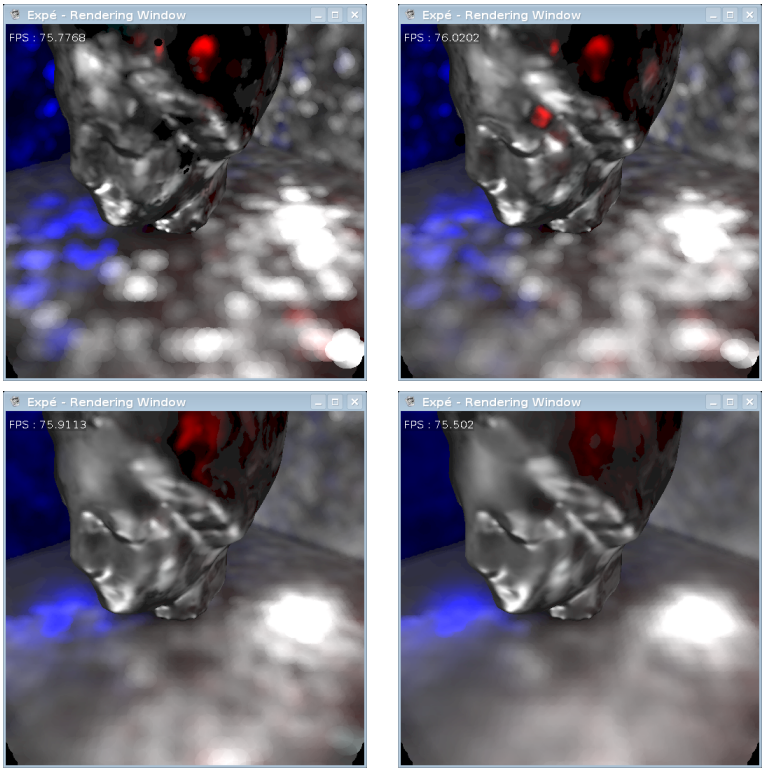

FIG. 4.8 – Comparaison du nombre de photons envoyés : 300 000 photons (209 317 photons reçus, 20.5 photons par splat en moyenne - 15 sec de lancer de photons, 7 sec d'optimisation), 600 000 photons (419 674 photons reçus, 41.6 par splat - 31 sec de lancer de photons, 14 secondes d'optimisation), 3 millions de photons (2 098 998 photons reçus, 209 par splat - 2 min 36 sec de lancer de photons, 1 min 13 sec d'optimisation), et 10 millions de photons (6 985 846 photons reçus, 698 par splat - 8 min 44 sec de lancer de photons, 4 min 19 sec d'optimisation).

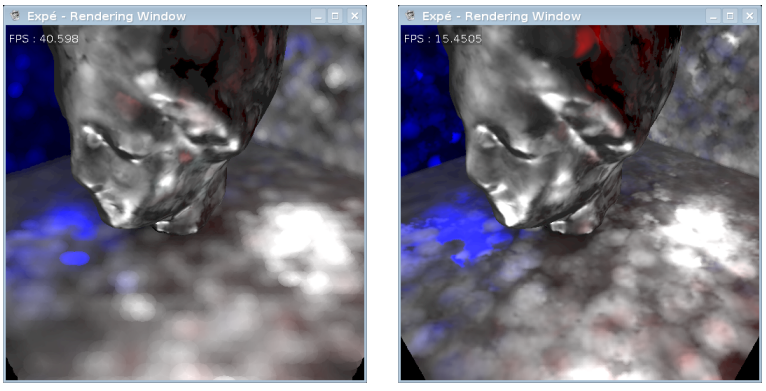

FIG. 4.9 – Comparaison du nombre de splats : 168 985 splats (Igea raffiné, 56 sec de lancer de photons, 53 sec d'optimisation), 398 305 (Igea et Cornell Box raffinés, 1 min 9 sec de lancer de photons, 2min 45s d'optimisation)

et l'intersection du photon avec les points inclus dans les voxels intersectés.
Ne possédant que peu de mesures (trois images en comptant la référence), nous ne nous prononcerons pas sur la complexité (qui ici semble être linéaire en nombre de splat, ce qui intuitivement devrait être le cas puisque chaque splat est indépendant).
Nous voyons que de hautes fréquences apparaissent lorsqu'il y a un nombre important de splats, ce qui s'explique par le fait que l'éclairage est considéré constant par splat ce qui peut éliminer les hautes fréquences si les splats sont trop gros.

4.3.4 en fonction de la taille de la zone de recherche

A chaque splat les photons sont cherchés dans un certain rayon lors de la phase d'optimisation. Nous avons évalué l'impact en temps de calcul et en qualité d'image de ce rayon de recherche (figure 4.10)
.

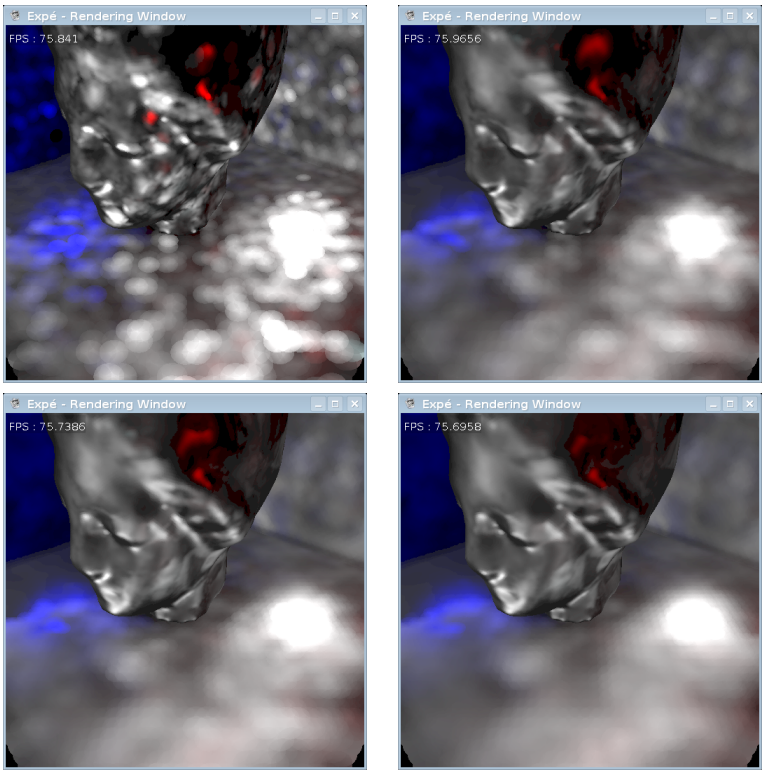

FIG. 4.10 – Comparaison de la taille de la zone de recherche : R=2 (11 sec d'optimisation, 30 photons par splat en moyenne) , R=4 (40 secondes d'optimisation, 123 photons par splat), R=5 (1min 2sec, 192 photons par splat), R=6 (1min 29 sec, 276 photons par splat)

A noter qu'un réétalonnage différent des luminosités a dû être opéré entre chaque image.
Le choix d'un rayon plus important a tendance à lisser le résultat. Un compromis entre le lissage "visuellement agréable" et la conservation des hautes fréquences liées à la spécularité des matériaux doit être trouvé.
Nous observons que la complexité est en $\mathcal{O}(R^2)$ ce qui est logique puisque l'aire de la zone de

recherche s'accroît en R^2.

4.3.5 en fonction de la géométrie

Nous proposons à titre qualitatif des rendus de différents modèles courants présentés figure 4.11

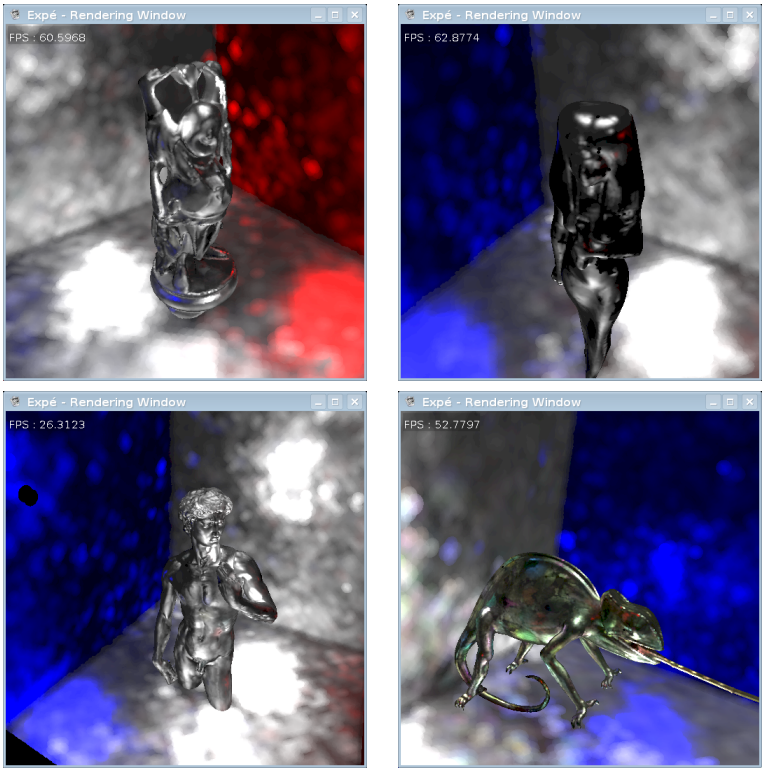

FIG. 4.11 – Bouddha (80 349 splats, 44sec de lancer de photons, 56sec d'optimisation), Isis (157 239 splats, 1 min 5 sec de lancer de photons, 32 sec d'optimisation), David (541 942 splats, 1 min 33 sec de lancer de photons, 5 min d'optimisation), Caméléon (198 191 splats, 1 min 01 sec de lancer de photons, 37 sec d'optimisation)

A noter les reflets colorés sur le caméléon qui est le seul modèle texturé (en nuances de vert pour son corps). Les images sont bruitées à cause du faible nombre de photons envoyés (toujours 1 million) et de la zone de recherche (toujours R = 3).
Une version plus lisse du caméléon est donnée à titre d'exemple (3 millions de photons, 5 min 8 sec de lancer de photons, 3 min 38 sec d'optimisation) en figure 4.12

4.3.6 avec l'éclairage direct

Nous n'avons pas encore, à l'heure du rapport, pu combiner un éclairage direct issu du moteur de rendu par point de Gaël Guennebaud par *Defered Shading* avec l'éclairage indirect issu

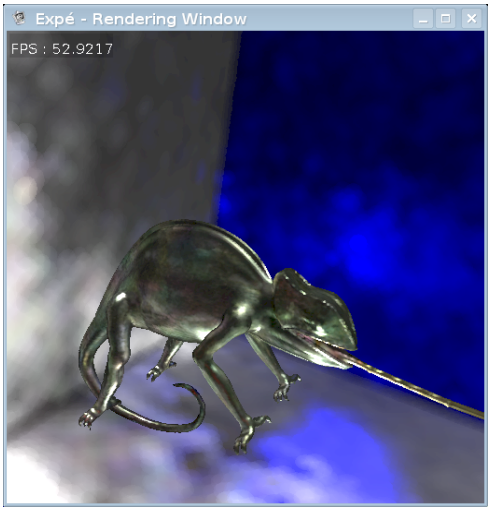

FIG. 4.12 – Caméléon lisse

de la solution par lancer de photons. Nous avons en revanche pu conserver les photons participant à l'éclairage direct (bien que ce ne soit pas une "bonne" solution puisque cela nécessite un nombre bien plus important de photons et donc un temps de calcul plus long), et ainsi obtenir une image dont l'éclairage direct est aussi modélisé par l'un des lobes gaussiens. Nous obtenons alors l'image 4.13.

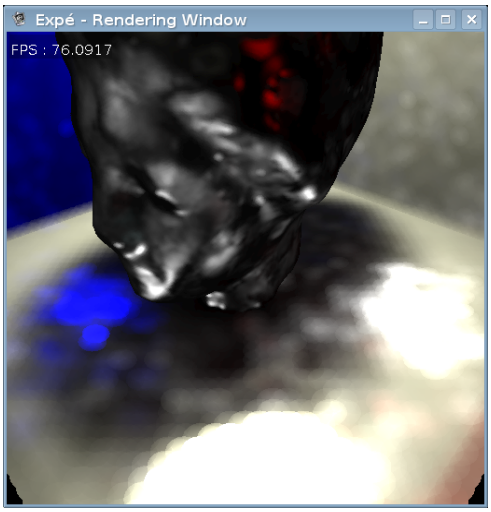

FIG. 4.13 – Igea avec éclairage direct

Cette image a été obtenue en 46 sec de lancer de photons et 7 min 11 sec d'optimisation : 1 million de photons ont été lancés, et 1 560 879 photons ont été reçus (dont ceux issus de l'éclairage direct). On a en moyenne 1 317 photons par splat.
A noter que l'utilisation d'un lobe pour simuler l'éclairage direct réduit la précision de l'éclairage indirect puisque l'éclairage indirect est modélisé avec un lobe de moins. De plus, la faible

absorption de ces matériaux (10% d'absorbtion) explique l'aspect artificiel de cette image. Un rendu plus complet (moins bruité...) est donné en figure 4.14.

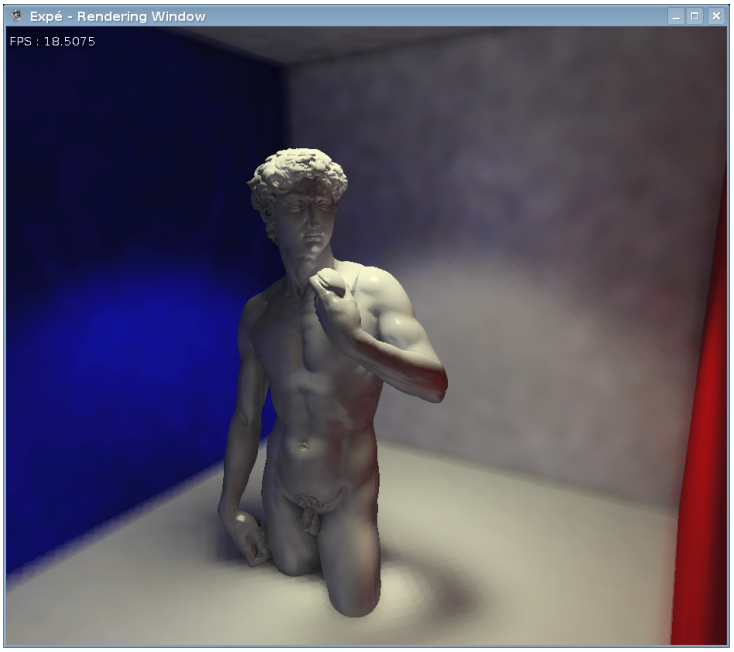

FIG. 4.14 – David avec éclairage direct

4.4 Temps d'affichage

Le nombre d'images par seconde (ou *fps* pour frame par second) est indiqué sur chaque image. Nous voyons que l'objectif de la reconstruction temps-réel est atteint puisque nous obtenons en moyenne entre 10 et 50 images par seconde. A noter que la phase de Tone Mapping (même si le Tone Mapping n'est pas appliqué à ces images, il est quand même calculé) réduit le débit d'affichage puisqu'une première passe doit être effectuée pour estimer le facteur de correction à appliquer aux luminances.
Globalement, l'estimation des luminances ainsi que la correction à appliquer par le Tone Mapping divise approximativement le débit d'affichage par deux comparé à la solution de l'éclairage direct seul.

5

Conclusion et perspectives

Nous avons pu développer un moteur d'éclairage global permettant de visualiser interactivement une scène statique prenant en compte des effets hautement spéculaires. Le précalcul effectué est beaucoup plus rapide que celui de nos prédécesseurs, mais le résultat semble moins précis et plus bruité à moins de diminuer les performances. Nous obtenons ainsi une simulation complète en moins de 5 minutes dans la plupart des cas, avec plus de 50 000 points alors que l'état de l'art mentionne généralement des calculs en plusieurs heures pour des modèles de moins de 10 000 points.
Une analyse de l'erreur commise sur l'approximation de la convolution par un lobe reste à faire, ainsi qu'un meilleur ajustement sur la somme des lobes.

L'hypothèse de la scène statique est très restrictive et des améliorations seraient à apporter afin de rendre la simulation complète en temps interactif. Une idée serait de combiner l'approche par lancer de photons interactif avec une estimation plus rapide des lobes.

Bibliographie

[AA03] A. Adamson and M. Alexa. Ray tracing point set surfaces, 2003.

[AH93] Larry Aupperle and Pat Hanrahan. A hierarchical illumination algorithm for surfaces with glossy reflection. *Computer Graphics*, 27(Annual Conference Series) :155–162, 1993.

[APS00] Michael Ashikmin, Simon Premo&#382 ;e, and Peter Shirley. A microfacet-based brdf generator. In *SIGGRAPH '00 : Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 65–74, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[BAOR06] Aner Ben-Artzi, Ryan Overbeck, and Ravi Ramamoorthi. Real-time brdf editing in complex lighting. In *ACM Transactions on Graphics (ACM SIGGRAPH 2006)*. ACM Press, 2006.

[CD91] Gilles Celeux and Jean Diebolt. Une version de type recuit simule de l'algorithme em. Technical report, January 1991.

[Cla03] Luc Claustres. *Modélisation de la Fonction de Distribution de la Réflectance Bidirectionnelle par Ondelettes pour le Rendu Physiquement Réaliste.* Thèse de doctorat, Université Paul Sabatier, Toulouse, octobre 2003.

[Cro] James L. Crowley. *Mélanges de Gaussiennes et L'Algorithme EM (Expectation-Maximization).*

[DÖ5] Arne Dür. On the ward model for global illumination. Unpublished material, 2005.

[DHS+05] Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan, and François Sillion. A frequency analysis of light transport. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2005)*, 24(3), aug 2005. Proceeding.

[DLWA] R. Dror, T. Leung, A. Willsky, and E. Adelson. Statistics of real-world illumination.

[DYN04] Yoshinori Dobashi, Tsuyoshi Yamamoto, and Tomoyuki Nishita. Radiosity for point-sampled geometry. In *Pacific Conference on Computer Graphics and Applications*, pages 152–159, 2004.

[GDW00] Xavier Granier, George Drettakis, and Bruce Walter. Simulation rapide de l'éclairage global. In J. Thollot and F. Neyret, editors, *Journees de l'AFIG 2000*. Association Française d'Informatique Graphique, 2000.

[GKMD06] Paul Green, Jan Kautz, Wojciech Matusik, and Frédo Durand. View-dependent precomputed light transport using nonlinear gaussian function approximations. In *SI3D '06 : Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 7–14, New York, NY, USA, 2006. ACM Press.

[GKPB04] Pascal Gautron, Jaroslav Křivánek, Sumanta N. Pattanaik, and Kadi Bouatouch. A novel hemispherical basis for accurate and efficient rendering. In *Rendering Techniques 2004, Eurographics Symposium on Rendering*, pages 321–330, June 2004.

[Gre03] Robin Green. Spherical harmonic lighting : The gritty details. In *Game Developpers' Conference*, 2003.

[HLF] Ondrej Hajdok, Antonín Lejsek, and Ondřej Fialka. Evaluation of tone mapping operators *http ://www.cgg.cvut.cz/ cadikm/tmo/.*

[IVD+04] A. Ignatenko, I. Valiev, K. Dmitriev, B. Barladian, S. Ershov, A. Voloboy, and V. Galaktionov. A real-time 3d rendering system with brdf materials and natural lighting. *Graphicon*, 2004.

[JMLH01] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan. A practical model for subsurface light transport. In *Proceedings of SIGGRAPH 2001*, pages 511–518, August 2001.

[Joz02] Timothy R. Jozwowski. Real time photon mapping. Master's thesis, Michigan Technological University, May 2002.

[KAMJ05] Anders Wang Kristensen, Tomas Akenine-Möller, and Henrik Wann Jensen. Precomputed local radiance transfer for real-time lighting design. *ACM Trans. Graph.*, 24(3) :1208–1215, 2005.

[KM00] Jan Kautz and Michael D. McCool. Approximation of glossy reflection with prefiltered environment maps. In *Graphics Interface*, pages 119–126, 2000.

[KMS05] Grzegorz Krawczyk, Karol Myszkowski, and Hans-Peter Seidel. Perceptual effects in real-time tone mapping. In *SCCG '05 : Proceedings of the 21st spring conference on Computer graphics*, pages 195–202, New York, NY, USA, 2005. ACM Press.

[Lew94] R. Lewis. Making shaders more physically plausible. 1994.

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

[NH98] R. Neal and G. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, 1998.

[NM03] Shu-Kay Ng and Geoffrey J. McLachlan. On some variants of the em algorithm for the fitting of finite mixture models. *Austrian Journal of Statistics, 32 :143-161.*, 2003.

[NRH03] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. All-frequency shadows using nonlinear wavelet lighting approximation. *ACM Trans. Graph.*, 22(3) :376–381, 2003.

[NRH04] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. Triple product wavelet integrals for all-frequency relighting. *ACM Trans. Graph.*, 23(3) :477–487, 2004.

[RSSF02] Erik Reinhard, Mike Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. *ACM Transactions on Graphics*, 21(3) :267–276, July 2002.

[SHHS03] Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph.*, 22(3) :382–391, 2003.

[SJ00] G. Schaufler and H. Jensen. Ray tracing point sampled geometry, 2000.

[SKS02] P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, 2002.

[SLS05] Peter-Pike Sloan, Ben Luna, and John Snyder. Local, deformable precomputed radiance transfer. *ACM Trans. Graph.*, 24(3) :1216–1224, 2005.

[TR93] Jack Tumblin and Holly Rushmeier. Tone reproduction for realistic images. *IEEE Comput. Graph. Appl.*, 13(6) :42–48, 1993.

[TS06] Yu-Ting Tsai and Zen-Chung Shih. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. In *ACM Transactions on Graphics (ACM SIGGRAPH 2006)*. ACM Press, 2006.

[Wal05] Bruce Walter. Notes on the ward brdf. Technical report PCG-05-06, Program of Computer Graphics, Cornell University, April 2005.

[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.

[WJ95] M. P. Wand and M. C. Jones. *Kernel Smoothing*, volume 60 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, London, 1995.

[WTL04] Rui Wang, John Tran, and David Luebke. All-frequency relighting of non-diffuse objects using separable BRDF approximation. In *Proceedings of the Eurographics Symposium on Rendering*, pages 345–354, 2004.

A

Comparaison des PDF de Duer et de Walter

Pour le modèle de BRDF de Ward isotrope, vérifions que les PDF (densités de probabilité) de Walter et de Duer sont les mêmes, à savoir : $pdf_{Walter} == pdf_{Duer}$

$$\begin{aligned}
\frac{\exp^{-\tan^2\delta/\alpha^2}}{4\pi\alpha^2(\vec{h}.\vec{i})\cos^3\theta_h} &=_? \frac{\exp^{-\tan^2\delta/\alpha^2}}{4\pi\alpha^2\cos\theta_i\ \cos\theta_r}.\frac{4\cos\theta_r(1+\cos\theta_i\ \cos\theta_r+\sin\theta_i\ \sin\theta_r\cos(\phi_r-\phi_i))}{(\cos\theta_i+\cos\theta_r)^3} \\
\Rightarrow \frac{1}{(\vec{h}.\vec{i})(\vec{h}.\vec{N})^3} &=_? \frac{1}{\cos\theta_i}.\frac{4(1+\cos\theta_i\ \cos\theta_r+\sin\theta_i\ \sin\theta_r\cos(\phi_r-\phi_i))}{(\cos\theta_i+\cos\theta_r)^3}
\end{aligned}$$

Posons : $R = 1+\cos\theta_i\ \cos\theta_r+\sin\theta_i\ \sin\theta_r\cos(\phi_r-\phi_i)$ comme dans [DÖ5].

Nous cherchons maintenant l'égalité :

$$\frac{1}{(\vec{h}.\vec{i})(\vec{h}.\vec{N})^3} =_? \frac{4}{\vec{N}.\vec{i}}.\frac{R}{(\cos\theta_i+\cos\theta_r)^3}$$

Nous avons :

$$\vec{h}.\vec{N} = \frac{\cos\theta_i+\cos\theta_r}{\sqrt{2R}}$$

puisque $\vec{h}$ est le vecteur médian entre $\vec{i}$ et $\vec{r}$.

Donc :

$$(\vec{h}.\vec{N})^3 = \frac{(\cos\theta_i+\cos\theta_r)^3}{2R\sqrt{2R}}$$

De plus, en utilisant les expressions en coordonnées sphériques des vecteurs $\vec{h}$ et $\vec{i}$ donnés dans [DÖ5], nous obtenons :

$$\begin{aligned}
\vec{h}.\vec{i} &= (\sin^2\theta_i\cos^2\phi_i+\sin\theta_i\sin\theta_r\ \cos\phi_i\cos\phi_r+\sin^2\theta_i\sin^2\phi_i \\
&\quad +\sin\theta_i\sin\theta_r\ \sin\phi_i\sin\phi_r+\cos^2\theta_i+\cos\theta_i\cos\theta_r)/\sqrt{2R} \\
&= \frac{\sqrt{R}}{\sqrt{2}}
\end{aligned}$$

Nous pouvons donc simplifier l'égalité que nous recherchons en :

$$\begin{array}{ccccc} \frac{\sqrt{2}}{\sqrt{R}}.\frac{2R\sqrt{2R}}{(\cos\theta_i+\cos\theta_r)^3} & =_? & \frac{4}{\vec{N}.\vec{i}}.\frac{R}{(\cos\theta_i+\cos\theta_r)^3} \\ \Rightarrow \quad 1 & \neq & \frac{1}{\vec{N}.\vec{i}} \end{array}$$

Nous voyons donc que l'égalité n'est pas vérifiée et que nous avons :

$$pdf_{Duer} = pdf_{Walter}\left(\vec{N}.\vec{i}\right)$$

Résumé

L'éclairage global consiste à calculer les multiples réflexions de la lumière dans une scène afin d'obtenir un certain réalisme lors de son rendu. Selon certaines restrictions et un précalcul long, des algorithmes permettent un rendu temps réel de l'éclairage global. Nous nous affranchissons ici de la restriction des matériaux basses fréquences (peu brillants) grâce à la modélisation de l'éclairage sous forme d'une somme de lobes Gaussiens, afin d'effectuer le rendu temps réel de scènes statiques avec un observateur dynamique. La simulation de l'éclairage est calculée par un lancer de photons, et l'ajustement des paramètres des lobes est effectué par un algorithme d'*Expectation-Maximization*. Une expression de l'équation du rendu est obtenue par une approximation de la convolution de lobes Gaussiens.

Mots-clés: Synthèse d'images, Éclairage global, BRDF, *Expectation-Maximization*

Abstract

The Global illumination purpose is to compute multiple reflections of light in a scene so as to get a certain realism when rendering. Considering some restrictions and a long precomputation time, some algorithms can render global illumination in real time. Our method avoids the restriction of low frequency materials (not very shiny) thanks to the lighting representation as a sum of Gaussian lobes, and allows to render static scenes in real time, with a moving viewer. The simulation of global illumination is done by a photon tracing step, followed by a lobe parameters fitting which is done by an *Expectation-Maximization* algorithm. A real-time evaluation of the rendering equation is achieved by approximating the convolution of Gaussian lobes.

Keywords: Image synthesis, Global illumination, BRDF, *Expectation-Maximization*