Université de Montréal

Placement automatique de sondes d'irradiance

par
Joël Polard-Perron

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

mai, 2017

RÉSUMÉ

L'illumination globale entre surfaces diffuses au sein d'une scène 3D s'approxime couramment en temps réel par l'utilisation de sondes d'irradiance. Le placement des sondes est une tâche qui requiert typiquement une importe intervention humaine, et la qualité du résultat final est souvent laissée au jugement de l'artiste d'éclairage. Comme la demande pour le réalisme en rendu augmente, le besoin de meilleures approximations émerge.

Nous proposons une méthode pour placer automatiquement des sondes dans une scène par minimisation d'une fonction d'erreur. Nous guidons les sondes vers les sites d'échantillonnage optimaux en appliquant la descente de gradient à une fonction d'erreur qui représente la similarité entre la structure en construction et un ensemble de référence.

En utilisant la pondération inverse à la distance comme fonction interpolante, nous avons construit avec fiabilité des ensembles de sondes dans trois scènes. En comparant nos résultats avec ceux produits par un ensemble de sondes de référence placées sur une grille régulière, nous atteignons théoriquement notre objectif dans une des trois scènes, où nous obtenons des valeurs d'erreur inférieures à la référence avec beaucoup moins de sondes. Nous avons eu des succès partiels dans les autres scènes, selon le nombre d'échantillons utilisés.

Mots clés: Irradiance, rendu, illumination globale, sondes lumineuses, interpolation, échantillonnage, optimisation.

ABSTRACT

Diffuse global illumination within a 3D scene can be approximated in real time using irradiance probes. Probe placement typically relies on significant human input, and final quality of the approximation is often left to the subjectivity of a lighting artist. As demand for realism in rendering increases, the need to enhance the quality of such approximations is greater.

We propose a method to automatically place probes in a scene by minimizing an error function. We guide probes to optimal sampling locations by applying gradient descent to an error function that represents similarity between our interpolated results and reference irradiance values.

Using weighted nearest neighbour interpolation, we were able to reliably construct probe sets with minimal input in three scenes. Comparing our results to those produced by a set of probes placed on a 3D grid, we were theoretically successful in one scene, in which we could obtain lower error values with fewer probes. We obtained partial success in the other scenes, depending on the number of samples used.

Keywords: Irradiance, Rendering, Global illumination, Light probes, Interpolation, Sampling, Optimisation.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

REMERCIEMENTS

Avant tout, merci à mes parents pour m'avoir appris la curiosité ; cet accomplissement n'aurait pas été possible sans leur support continuel. Merci à Marie-Élaine pour sa présence tout au long de mon cheminement ; grâce à toi, les périodes plus sombres étaient toujours bien éclairées. Merci à tous les amis du LIGUM, avec qui j'ai eu le plaisir de partager notre petit espace de laboratoire. Finalement, merci à Derek, qui m'a offert une chance en or en me donnant l'opportunité de me joindre à son équipe.

CHAPITRE 1

INTRODUCTION

1.1 Rendu en infographie

L'*infographie*, en tant que domaine sous-jacent à l'informatique, désigne l'étude et l'application de procédés par lesquels on peut produire des images grâce aux ordinateurs. Ce vaste champ d'études englobe un grand nombre de composantes qui agissent ensemble dans un but commun : produire une image à partir d'une gamme d'informations décrivant une scène. Cette description de scène inclut une variété de paramètres ; notons par exemple les conditions d'éclairage, les surfaces géométriques, les matériaux sur ces surfaces, etc.

Figure 1.1 – Scène générée par ordinateur du film *Brave* (2012), de Pixar [14].

De plus en plus, le but visé par les créateurs d'images est le réalisme : les images produites doivent être conformes à la réalité. Les phénomènes visuels présents dans notre monde découlent principalement des interactions entre la lumière, les surfaces et les volumes participants (par exemple, le brouillard) ; ceux-ci doivent donc être compris, modélisés, puis simulés à l'aide d'algorithmes. Le transport de lumière dans le monde réel est un phénomène très complexe qui n'est pas encore entièrement compris

par la communauté scientifique ; il trouve ses racines au sein de la physique quantique, un domaine d'études très spécialisé qui se situe bien au-delà du sujet de ce mémoire et qui nécessite un traitement mathématique élaboré. L'infographie se concentre principalement sur les phénomènes macroscopiques du transport de la lumière ; nous nous contenterons généralement d'approximations qui peuvent être simulées grâce à des outils mathématiques plus simples que la description quantique.

Les phénomènes contribuant à une représentation réaliste du transport de lumière sont regroupés sous la bannière de l'*illumination globale* (ou éclairage global). Ce terme est défini en opposition à l'*illumination directe*, qui ne considère que les sources directement visibles de lumière et ne peut donc représenter qu'un sous-ensemble des phénomènes réels. L'illumination globale ajoute les sources indirectes d'éclairage, ce qui correspond mieux à la réalité puisqu'un rayon de lumière peut avoir rebondi sur plusieurs surfaces dans une scène avant d'arriver finalement à l'appareil de capture. Au niveau quantique, chaque rebond correspond à l'absorption par la surface du photon, puis généralement à l'émission d'un autre photon dont le spectre est altéré selon les propriétés de la surface. La lumière ayant une vitesse très grande, ce processus se passe tellement vite que nous ne pouvons qu'observer le résultat final sur l'ensemble des surfaces, que nous considérons comme étant *à l'équilibre*. Ce processus d'absorption suivi d'une émission fait que chaque surface devient une source d'éclairage, que nous qualifierons d'*indirecte* puisque ces surfaces ne sont pas des émetteurs primaires. Ces sources de lumière indirectes contribuent à une grande partie de la lumière visible dans une scène réelle ; elles créent également des effets d'interréflexion entre les surfaces. La figure 1.2 offre un exemple de comment l'utilisation d'algorithmes d'illumination globale améliore la qualité visuelle d'un rendu : l'image comportant seulement les composantes directes présente moins de détails intéressants.

Nous pouvons diviser les différentes techniques de rendu en deux catégories : le rendu *en ligne* et le rendu *hors-ligne*. Ces deux catégories diffèrent de par le temps jugé acceptable de production d'une image. Le rendu hors-ligne est présent, par exemple, dans les industries du cinéma ou de l'architecture, où chaque image est générée avant que le public ne la visionne. Les paramètres formant chaque image sont déterminés

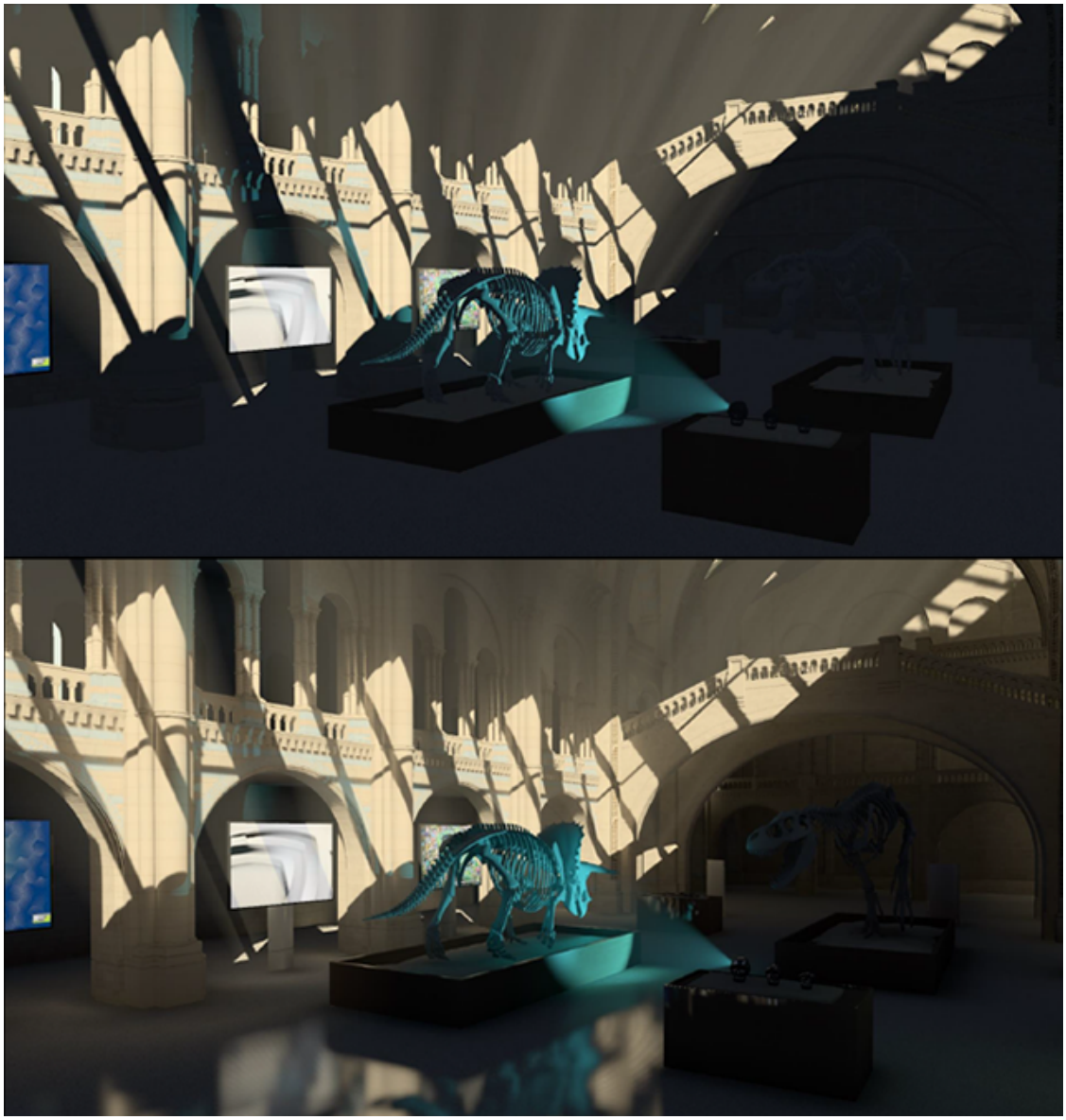


Figure 1.2 – Comparaison de rendu sous éclairage direct (haut) et sous éclairage global (bas) [7].

d'avance par les artistes qui les créent. Ainsi, chaque image peut être travaillée et polie au maximum, et les techniques utilisées pour simuler les phénomènes à représenter peuvent être aussi coûteuses que nécessaires et techniquement possible, autant en temps qu'en utilisation de ressources informatiques. Ainsi, dans ce domaine, il n'est pas rare que les temps de rendu atteignent plus de 15 heures par image, en plus du temps consacré à la

modélisation qui peut prendre des jours, voire des semaines.

À l'autre bout du spectre, le rendu en-ligne vise à produire des images très rapidement, afin d'offrir une expérience interactive au client. L'industrie du jeu vidéo est un bon exemple. Les jeux vidéo laissent généralement une grande part de liberté à l'utilisateur, et ceci peut inclure la liberté de contrôler la position et l'orientation de la caméra virtuelle présente dans le jeu. Il devient alors nécessaire, pour donner l'impression du mouvement, de générer les images successives à haute fréquence, à la manière d'une pellicule vidéo. Les jeux vidéo tentent généralement de produire des images à une fréquence variant entre 30 et 60 Hz, selon la fluidité désirée et la puissance de calcul disponible.

Figure 1.3 – Le jeu *Quantum Break* (2016), de Remedy Entertainment [25].

Pour atteindre ces vitesses de rendu, divers leviers sont utilisés, tels que des algorithmes sacrifiant le réalisme du résultat au profit de la rapidité d'exécution, ainsi que l'utilisation de matériel spécialisé pour le traitement d'informations graphiques, tel que les cartes graphiques présentes dans la plupart des ordinateurs et des consoles de nos jours.

1.2 Notre travail

Nous nous pencherons dans ce mémoire sur une méthode couramment utilisée pour approximer l'irradiance provenant de sources indirectes sur l'ensemble d'une scène, soit les *sondes d'irradiance*. Les sondes échantillonnent la fonction sphérique d'irradiance à des points discrets dans la scène et permettent, par interpolation, d'approximer l'irradiance incidente sur les surfaces. Ceci permet d'estimer rapidement le terme d'illumination indirecte sur chaque pixel de l'image finale : plutôt que de calculer le terme d'irradiance incidente sur chaque élément de surface visible dans l'écran, une opération qui consommerait une portion substantielle du temps alloué au rendu, on en pré-calcule une approximation lors de la production du jeu, ce qui permet le précalcul. On stocke ensuite l'approximation de cette irradiance sous une forme compacte, bien adaptée à une évaluation rapide.

Cette méthode repose sur la supposition qu'il est possible d'obtenir une estimation satisfaisante de l'irradiance incidente à une surface en l'interpolant à partir d'une représentation volumétrique. Il importe donc d'obtenir la représentation volumétrique de la fonction d'irradiance incidente la plus fidèle possible à la réalité. La solution offrant la reconstruction la plus précise serait de stocker les valeurs d'irradiance incidente en tout point de l'espace et pour toutes les directions ; tout élément de surface pourrait aller directement chercher la valeur lui correspondant dans la table. Cette solution n'est pas réalisable sur le plan pratique, puisqu'il faudrait échantillonner un nombre immense de points, ce qui nécessiterait un temps et une capacité de stockage démesurés.

En pratique, on se tournera donc vers des stratégies d'échantillonnage qui tirent profit du fait que l'irradiance est une fonction variant généralement lentement dans l'espace. On pourra donc, par exemple, séparer le volume englobant la scène en cellules, telles que celles formées par une grille régulière en trois dimensions, et interpoler linéairement sur les surfaces au sein des cellules en utilisant les valeurs stockées aux sommets de la grille.

La sélection des points auxquels l'irradiance est échantillonnée et stockée a une influence directe sur la qualité du résultat final. Si un élément de surface utilise un ou des échantillons contenant des valeurs d'irradiance inappropriées, le *shading* obtenu ne

sera pas plausible, allant à l'encontre du but de réalisme fixé par les applications 3D modernes.

Il importe donc de porter une attention particulière à la sélection des points utilisés pour l'échantillonnage, puisqu'elle implique une plus ou moins grande qualité du résultat final, dont le jugement est en général mieux laissé aux soins d'un humain.

On se retrouve donc devant une situation où nous devons équilibrer plusieurs facteurs. On veut obtenir la meilleure interpolation possible de l'irradiance pour un point arbitraire au sein de la scène. Il faut limiter le nombre d'échantillons utilisé, puisque leur génération et leur stockage risquent de consommer une grande quantité de ressources. Idéalement, on souhaite que le travail soit automatisé, puisque la génération de la structure d'approximation et l'évaluation du résultat final est un travail qui requiert du temps et peut offrir un résultat variable si laissé au seul jugement d'un artiste d'éclairage.

D'autre part, les ressources disponibles aux créateurs de scènes sont limitées, et la nature itérative du processus fait qu'un artiste peut avoir à retravailler plusieurs fois une même scène au cours de son évolution. Un placement de probes automatisé permettrait donc de gagner du temps aux créateurs de scènes.

Une méthode automatique de placement des probes doit produire un résultat correct. Nous formulons le problème sous la forme d'un problème d'optimisation numérique : nous définissons une valeur d'erreur exprimant la grandeur de la différence entre notre reconstruction de l'irradiance et une reconstruction optimale, et utilisons le gradient de cette fonction pour trouver les emplacements correspondant à des valeurs minimum.

Pour atteindre l'objectif, plusieurs facteurs sont à considérer. Il faut choisir une fonction appropriée exprimant la valeur que nous voulons minimiser. Ceci implique le choix d'une méthode pour interpoler les échantillons ; nous avons mentionné la grille régulière, mais d'autre choix pourraient être appropriés. Il faut également choisir une métrique selon laquelle comparer les ensembles d'échantillons à comparer.

L'algorithme doit s'appliquer sur une scène arbitraire avec un minimum d'intervention humaine et converger vers une solution optimale à chaque exécution, autrement, il ne pourrait être utilisable dans une production professionnelle. Il doit également converger *rapidement*. et produire un résultat consistent ; autrement dit, différents ensembles

de sondes produisant la même valeur d'erreur doivent produire un résultat final perceptuellement indistinguable.

Nous avons eu un succès partiel dans la réalisation de ces objectifs. La méthode que nous avons développée ne fonctionne pas dans toutes nos scènes de test, mais nous avons tout de même des résultats positifs : Nous avons réussi à construire sans intervention des structures de probes et à faire baisser la valeur de notre fonction d'erreur en-deçà de celle produite par des échantillons placés manuellement sur une grille dans une de nos scènes, ce qui suggère que l'objectif est atteignable, mais la qualité de la reconstruction des échantillons générés n'est pas satisfaisante. Les divers critères mentionnés précédemments, soit la représentativité de la fonction d'erreur, la qualité de la reconstruction, la diversité des scènes applicables et l'optimisation du nombre d'échantillons requis sont autant de critères mal remplis par notre méthode.

Une description complète de notre méthode et de ses résultats requiert une compréhension approfondie des différents éléments formant le système. Le prochain chapitre établira donc la base de la théorie en formalisant le transport de la lumière, puis en décrivant le fonctionnement des sondes.

Nous verrons ensuite l'état de l'art en construction automatique de structures de sondes, avant de décrire notre méthode, puis de présenter nos résultats.

CHAPITRE 2

THÉORIE

Avant de présenter notre méthode, il importe de montrer les bases théoriques à partir desquelles elle est construite. Nous présenterons donc d'abord les concepts mathématiques nécessaires afin de former un cadre dans lequel enchâsser notre travail.

2.1 Radiométrie

Le point de départ de notre travail est la *radiométrie*, l'étude du rayonnement électromagnétique. La production d'images synthétiques réalistes nécessite une compréhension des concepts de base de ce domaine qui relève de la physique.

La lumière visible est composée d'un très grand nombre de photons, de longueurs d'ondes différentes formant un spectre allant de 400 à 700 nm. Une source de lumière de puissance P émet continuellement des photons, dans toutes les directions, à partir de chaque point de sa surface. Le spectre d'émission varie selon divers paramètres inhérents à la source ; par exemple, on peut catégoriser les étoiles en différentes catégories selon leur spectre d'émission, qui sera différent du spectre émis par une ampoule incandescente [30].

Les photons transportent une certaine quantité d'*énergie électromagnétique* ; la quantité d'énergie qu'un photon transporte est une fonction de sa longueur d'onde. Le transport d'énergie sur une période de temps continu nous amène à parler de *puissance électromagnétique* ; en radiométrie, on appelle *flux radiant* la quantité d'énergie émise par une source en fonction du temps.

Toutes les surfaces recevant des photons les absorbent, puis émettent à leur tour de nouveaux photons. On distingue donc les émetteurs en sources *primaires* et *secondaires*. Générer une image, soit par ordinateur ou avec une caméra, revient à capturer la *radiance* sortante émise par les surfaces en direction de l'appareil de mesure. La radiance est définie comme étant la puissance passant par une unité de surface vers une unité d'angle

solide [18] ; l'accumulation de la radiance incidente, soit dans les senseurs de la caméra ou les pixels d'une image virtuelle, va constituer l'image finale.

2.2 Irradiance

La quantité de radiance émise par une source secondaire dépend en partie de l'*irradiance* reçue par la surface. L'irradiance est la puissance reçue par une surface par unité d'aire. Chaque point d'une surface reçoit de la radiance provenant de toutes les directions ; l'intégrale de la radiance provenant de chacune de ces directions sur un hémisphère centré au point $\vec{x}$ autour de la normale $\vec{n}$ de la surface correspond à l'irradiance [18] :

$$E(\vec{x}) = \int_{\Omega} L(\vec{x}, \omega) \lfloor \omega \cdot \vec{n} \rfloor d\omega_i. \tag{2.1}$$

On note que les termes contenus dans l'intégrale sont pondérés par un facteur appelé le cosinus tronqué. Ce terme provient du fait que plus l'angle entre une surface et la direction du flux incident est grand, plus le flux se fait répartir sur un plus grand angle solide sur l'hémisphère d'intégration, ce qui affaiblit le flux reçu par unité d'aire sur la surface. La figure 2.1 est une représentation schématique de la sommation hémisphérique de l'irradiance entrante au point $\vec{x}$.

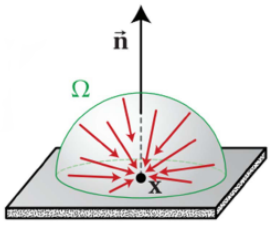


Figure 2.1 – Irradiance [29].

2.3 BRDF

Après absorption, une partie de l'énergie reçue par la surface est réémise sous la forme d'un photon de longueur d'onde potentiellement différente [30]. Cette *radiance sortante* n'est pas nécessairement émise uniformément dans toutes les directions ; en fait, c'est rarement le cas dans le monde réel. Toutes les surfaces réelles présentent des imperfections, soient accidentelles ou délibérées, au sein de leur micro-géométrie, qui

font que certaines directions seront privilégiées lors de l'émission ou de la réémission de radiance. La *BRDF* (Bi-Directional Reflection Distribution Function) $f_r(\omega_i, \omega_o)$ d'une surface représente la quantité de lumière qu'elle réfléchit d'une direction ω_i vers une direction ω_o [39]. Celle-ci est schématisée à la figure 2.2 ; la distance du pointillé au point $\vec{x}$ représente la probabilité qu'un rayon soit dévié dans la direction correspondante.

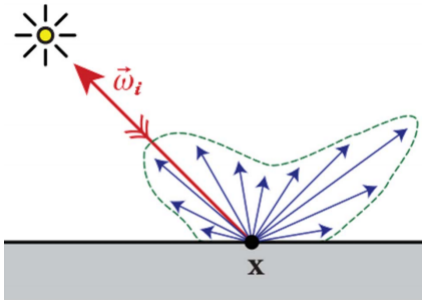


Figure 2.2 – BRDF [29].

L'obtention de la BRDF exacte d'une surface réelle est difficile. On procède généralement par échantillonnage de la réponse lumineuse d'une surface à divers angles d'illumination incidente et réfléchie. Les données obtenues peuvent alors être ajustées à des modèles mathématiques plus ou moins complexes. L'industrie du rendu utilise couramment, de nos jours, le modèle de microfacettes [22]. Trouver un bon modèle est un choix qui nécessite la prise en compte de plusieurs variables : la complexité du calcul requis pour le simuler est une composante importante, et des applications en temps réel peuvent rarement se permettre de sacrifier du temps de calcul pour une augmentation de qualité difficile à percevoir. Il est également important que le modèle soit simple à utiliser par les artistes qui créeront les matériaux présents dans les scènes à éclairer ; un modèle plus précis ou général mais qui requiert d'ajuster finement un grand nombre de paramètres abstraits risque d'être moins utilisable que des modèles plus simples.

Notons finalement le modèle de BRDF le plus simple que l'on peut utiliser, soit le modèle diffus (aussi appelé *Lambertien*) ; c'est celui que nous avons utilisé pour la génération de probes au cours de notre recherche. Le *shading* (nom donné au processus d'évaluation de la couleur d'une surface) spéculaire est dépendant du point de vue, ce qui est incompatible avec l'échantillonnage statique des probes. La BRDF diffuse est simplement celle qui renvoie l'irradiance qu'elle reçoit uniformément dans toutes les directions :

$$f_r(\omega_i, \omega_r) = \frac{\rho}{\pi},$$

où ρ est l'*albedo* de la surface ; cette quantité donne la quantité de lumière réfléchie pour chaque composante du spectre lumineux. Pour une surface entièrement rouge, par exemple, l'albedo dans l'espace *RGB* sera $\rho = (1,0,0)$. L'énergie lumineuse renvoyée étant uniforme dans toutes les directions, le *shading* ne changera pas selon le point de vue [29].

2.4 Équation de rendu

Ces outils mathématiques nous permettent maintenant de présenter l'*équation de rendu*, une équation très importante en infographie moderne [24] :

$$L_o(\vec{x}, \omega_o) = L_e(\vec{x}, \omega_o) + \int_\Omega L(\vec{x}, \omega) f_r(\omega_i, \omega_o) \lfloor \omega_i \cdot \vec{N} \rfloor d\omega_i. \tag{2.2}$$

Cette équation nous indique que la radiance entrante $\vec{L}$ provenant d'un point dans la direction ω_o donnée est égale à la somme de :

- la radiance émise par la surface en direction de ω_o
- l'irradiance entrante à cette surface, pondérée par la BRDF et un facteur cosinus évalués pour toutes les directions de l'intégrale sur l'hémisphère.

Une représentation schématique de cette opération se trouve à la figure 2.3. La lumière qui arrive des directions ω_i dans l'hémisphère Ω est multipliée par la BRDF et le terme cosinus ; la somme de toutes ces contributions est renvoyée à l'observateur de $\vec{x}$ vers ω_o.

Il s'agit précisément de ce qu'il nous faut : une image n'est qu'un échantillonnage de la radiance entrante dans une série de capteurs. L'évaluation de la fonction de rendu pour des rayons entrant dans le capteur d'une caméra virtuelle est alors le coeur du rendu réaliste.

La difficulté réside dans le fait que la fonction est récursive. La radiance sortant d'un élément de surface situé à $\vec{x}$ dépend de l'irradiance y entrant ; pour évaluer cette irradiance, on doit considérer tous les éléments de surface dans la scène envoyant de la

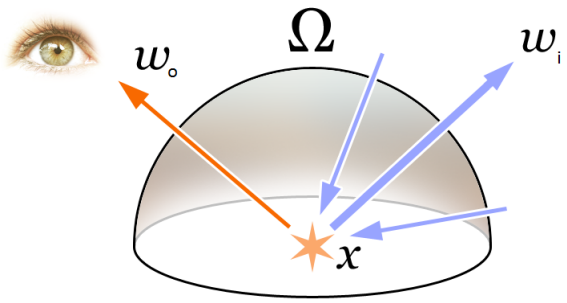


Figure 2.3 – Représentation schématique de l'équation de rendu [29].

radiance sur $\vec{x}$. La radiance sortant de ces points dépend à son tour de l'irradiance entrant à $\vec{x}$, ce qui nous plonge dans une boucle récursive infinie.

Une difficulté supplémentaire réside dans le fait que pour obtenir une évaluation exacte de la fonction, *tous* les éléments de surface visibles à partir de chaque point dont on veut connaitre l'irradiance incidente doivent être considérés. Ceci mène très rapidement à une explosion du nombre d'éléments de surface à considérer. L'approche naïve menant à ce problème, on doit utiliser une approche plus sophistiquée.

2.5 *Path tracing*

Kajiya, en plus de définir l'équation de rendu, a introduit une méthode ingénieuse pour la résoudre, soit le *path tracing*, qui est une forme d'intégration *Monte Carlo*. Ce nom réfère à une classe de méthodes qui incluent l'estimation d'une intégrale de façon probabiliste. Une méthode Monte Carlo remplace une équation du type

$$y = \int f(x)dx$$

par l'approximation suivante :

$$y = \frac{1}{N}\sum_{i}^{N}\frac{f(x_i)}{\text{pdf}(x_i)},$$

où les x_i sont des points choisis aléatoirement sur le domaine d'intégration [3]. $\text{pdf}(x_i)$ est la *fonction de densité de probabilité* de l'échantillon x_i, représentant la probabi-

lité que l'échantillon x_i ait été choisi relativement à un autre point sur le domaine d'intégration. Le processus introduit inévitablement une erreur dans la valeur calculée, mais des échantillons soigneusement choisis peuvent réduire la valeur de cette erreur. L'*importance sampling* est le nom donné à la sélection d'échantillons fournissant de meilleures informations que d'autres ; on préfèrera échantilloner aux endroits où on risque de trouver la meilleure information possible sur la fonction.

Le *path tracing* évalue l'équation de rendu en simulant le transport de lumière à l'envers de la direction réelle ; il procède en envoyant, pour chaque pixel de l'image à évaluer, un rayon lumineux virtuel qui sort de la caméra en direction de la scène. Si ce rayon touche une surface de la scène, l'intersection est stockée, et un nouveau rayon est envoyé dans une direction aléatoire. Le processus est répété jusqu'à ce que le rayon touche une surface émettant de la radiance ; on a alors trouvé un chemin de lumière contribuant à la radiance totale entrant dans le pixel. Le processus est illustré à la figure 2.4. La répétition du processus un grand nombre de fois correspond à l'évaluation de l'intégrale de rendu via Monte Carlo. Les lobes au dessus de chaque sphère représentent les directions que peut prendre le rayon réfléchi. La distance de chaque point sur les lobes au point de rebond représente la probabilité que cette direction soit choisie ; autrement dit, c'est une schématisation de la BRDF.

Le *path tracing* est définitivement une méthode hors-ligne de rendu d'image. Pour une scène typique, le nombre de chemins de lumière contribuant à l'image finale est incommensurable ; le nombre d'échantillons requis pour obtenir une contribution de la part de chacun d'entre eux est immense. Ce problème se manifeste généralement sous la forme de *bruit* présent sur l'image finale, visible comme une variation perceptible de la radiance accumulée sur des pixels adjacents. Il importe alors d'appliquer des algorithmes de débruitage, ou d'augmenter drastiquement le nombre de rayons lancés. Il n'est pas rare, pour obtenir une image avec moins de bruit, qu'il faille lancer des centaines de milliers de rayons pour chaque pixel, ce qui est à l'origine des temps de rendu disproportionnés des images générées hors-ligne. Par exemple, pour *Big Hero 6*, Disney a dû distribuer le rendu des images du film sur 55 000 coeurs physiques [38], chaque image prenant plusieurs heures. Une application soigneuse de la méthode donne cepen-

dant des résultats impeccables, tel que celui à la figure 2.5.

2.6 Rastérisation

On peut séparer les méthodes de création d'images en deux paradigmes. Le premier, qui inclut le *path tracing* vu au chapitre précédent, génère les images par évaluations successives de la fonction de radiance entrant dans chaque pixel. Les applications en temps réel utilisent très rarement cette méthode, puisque le nombre d'échantillons requis pour obtenir un résultat non bruité cause des temps de rendu bien supérieurs à ce qui est nécessaire pour offrir une interactivité satisfaisante à l'utilisateur.

Les applications en temps réel découplent généralement les conditions d'éclairage de la géométrie d'une scène. On procède en deux phases : on *rastérise* d'abord la géométrie contre le plan image, puis on applique un algorithme de *shading* sur chaque élément de surface contribuant à l'image finale [15].

La rastérisation est une méthode projective qui sert à marquer les pixels d'un écran touchés par la géométrie d'une scène. Le processus est si courant qu'il est maintenant accéléré matériellement ; en effet, toutes les cartes graphiques modernes ont du matériel dédié qui applique la rastérisation de chaque triangle sur le plan image, garantissant une exécution très rapide. Une carte graphique opère en une série d'étapes successives ; certaines de ces étapes sont programmables (*shaders*) et les autres sont fixées, effectuant une opération bien précise.

Pour rastériser une scène 3D, on la transforme d'abord pour que chaque sommet soit exprimé en *espace caméra*. Ceci se fait à l'étape du *vertex shader*. Bien qu'il soit programmable et donc capable d'opérations exotiques, le *vertex shader* classique prend en entrée une série de sommets et les transformations requises pour l'exprimer en espace caméra.

Ces transformations sont encodées sous forme de matrices ; l'application d'une matrice à un vecteur (dans ce cas, la position du sommet) correspond à une *transformation linéaire* qui fait passer le vecteur d'une base à une autre. Dans ce cas, le sommet passe de l'*espace objet* à l'*espace écran* par une série de transformations.

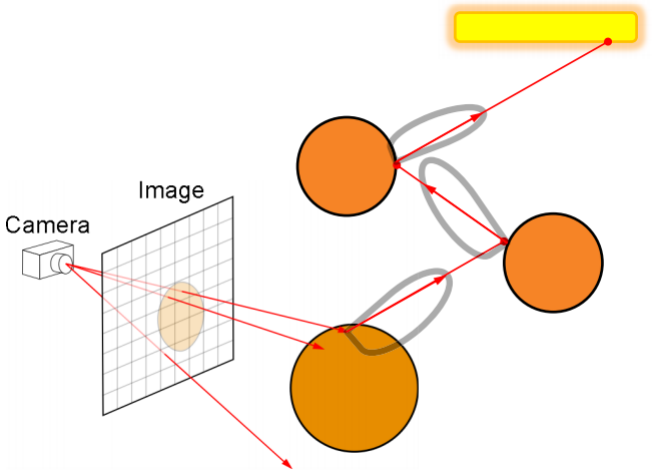


Figure 2.4 – Schématisation du processus de path tracing [29].

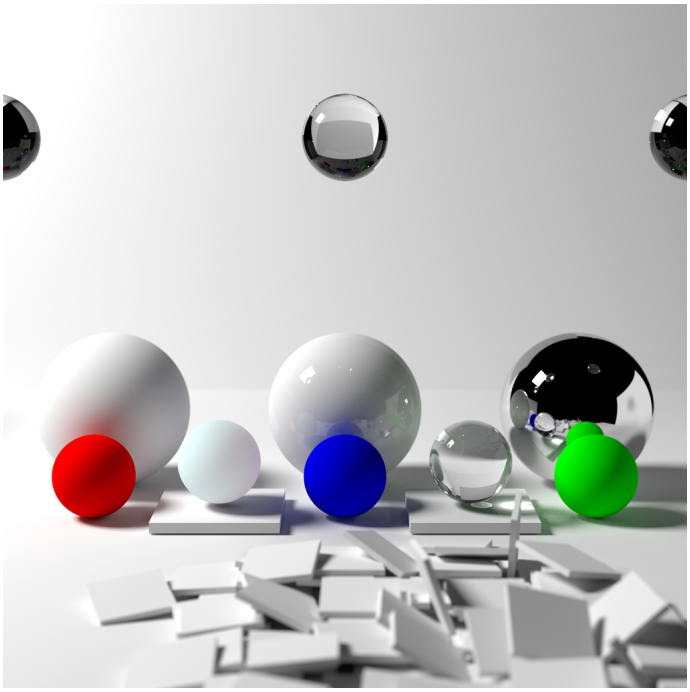


Figure 2.5 – Exemple d'image produite par *path tracing* [6].

Les coordonnées en espace écran de chaque sommet sortent du *vertex shader* et passent par le *pipeline* de la carte graphique ; celui-ci se charge de trouver les triangles visibles, de faire la *tessellation* des surfaces (c'est-à-dire de découper les surfaces en davantage de triangles) et d'interpoler les variables propres aux sommets (normales, couleurs, coordonnées de textures) sur les faces des triangles.

Finalement, le *fragment shader* est la dernière étape programmable, et c'est là qu'on retrouve les algorithmes de *shading* qui nous intéressent. Sa tâche est de prendre des *fragments* en entrée et de leur assigner une couleur ; essentiellement, de résoudre l'équation de rendu pour l'élément de surface du fragment. Une panoplie d'algorithmes ont été développés pour le *shading* des fragments ; cette étape est très flexible et on pourrait carrément faire du *path tracing* dans le *shader*, mais ce serait extrêmement inefficace.

Une façon courante de procéder est plutôt de tirer avantage du fait que la radiance est additive, et de séparer les contributions [17]. Ceci permet d'exprimer le résultat final comme la somme de termes calculés indépendamment. On distingue donc les contributions provenant de sources *directes* ou *indirectes*. Ces deux termes peuvent se calculer de façons complètement différentes. Soit un élément de surface à $\vec{x}$ dont la normale est $\vec{n}$; en prenant $\Omega_d(\vec{x})$ comme l'ensemble des directions pointant de $\vec{x}$ vers les sources directes, et $\Omega_i(\vec{x})$ comme l'ensemble des directions pointant vers des éléments de surface n'émettant pas directement de radiance, on aura :

$$L_e(\vec{x},\omega_o) = L_e(\vec{x},\omega_o) + \int_{\Omega_d(\vec{x})} F(\vec{x},\omega,\omega_o,\vec{s})d\omega + \int_{\Omega_i(\vec{x})} F(\vec{x},\omega,\omega_o,\vec{n})d\omega$$

avec

$$F(\vec{x},\omega,\omega_o,\vec{n}) = L_e(x,\omega) f_r(\vec{x},\omega,\omega_o) \lfloor \omega \cdot \vec{n} \rfloor.$$

Une représentation schématique de cette séparation des contributions se trouve à la figure 2.6.

Les contributions provenant de certaines formes de sources directes peuvent se calculer analytiquement. Par exemple, si la source est une *lumière ponctuelle*, on peut précisément calculer la radiance qu'elle envoie directement sur un élément de surface :

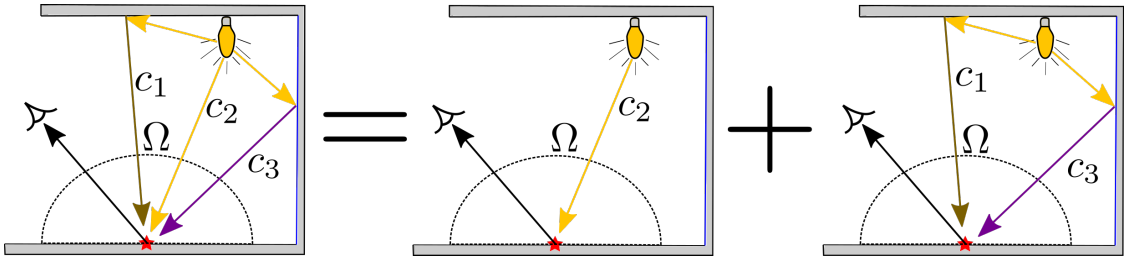


Figure 2.6 – Séparation des contributions en sources directes et indirectes.

une source ponctuelle émet dans toutes les directions uniformément, et sa radiance émise à une distance d varie selon l'inverse du carré de d. La radiance directe L_d reçue par cet élément de surface pour une lumière ponctuelle d'intensité I située à $\vec{p}$ sera donc

$$L_d = \frac{I(\vec{N} \cdot \vec{D})}{||\vec{p} - \vec{x}||^2}.$$

où $\vec{D}$ est le vecteur $(\vec{p} - \vec{x})$ normalisé.

Les moteurs de rendu offrent diverses formes de sources calculables analytiquement. Des méthodes géométriques permettent d'obtenir une approximation de la lumière émise par une forme polygonale arbitraire, si elle est entièrement visible du point à *shader*. Une fois de plus, la radiance étant additive, on peut simplement exécuter le *fragment shader* sur le même objet pour chaque source de lumière l'affectant et additionner les résultats [21].

La radiance de sources indirectes, quant à elle, n'a généralement pas de solution analytique. Une méthode typique pour la calculer est par l'utilisation d'une carte d'environnement.

2.7 Cartes d'environnement

La génération de cartes d'environnement a plusieurs utilités ; elles peuvent servir à l'évaluation des réflexions sur les surfaces spéculaires de la scène, ainsi qu'à l'évaluation de l'irradiance provenant de sources indirectes [27].

Une carte d'environnement est simplement un rendu de la scène pour toutes les directions autour d'un point. Elles sont représentables sous plusieurs formes : on trouve

par exemple les *cubemaps*, qui sont 6 rendus de la scène selon les 6 axes principaux ($\pm$ x, y et z), ou les cartes *équisphériques*, qui encodent toutes les directions sur une image rectangulaire. On peut voir un exemple classique à la figure 2.7.

L'utilisation de cartes environnementales est une forme de *précalcul* ; on entend par ici que des informations utilisables en temps réel sont calculées et stockées d'avance dans les données de la scène.

Très tôt en rendu, les cartes d'environnement étaient utilisées pour approximer les réflexions spéculaires [1]. Ces réflexions exigent d'aller chercher la radiance entrante le long du vecteur opposé (miroir) au vecteur d'oeil. Si l'environnement est considéré comme étant à une grande distance du point de *shading*, on peut simplement aller chercher la radiance sur la carte dans la direction correspondante. La supposition que l'environnement est loin fait qu'un déplacement dx du point de *shading* ne changera pas le texel (élément de texture) à évaluer sur la carte environnementale.

Pour des réflexions *glossy*, les directions à évaluer se situent au sein d'une forme (généralement une gaussienne de directions, appelée un *peak de Phong*). Ces réflexions peuvent alors être approximées en faisant un *préfiltrage* de la carte avec un noyau équivalent à la forme à échantillonner, selon la BRDF. Les texels filtrés contiennent alors un moyennage des texels environnants, simulant l'effet d'une réflexion *glossy*.

Les cartes environnementales sont également couramment utilisées pour évaluer la fonction sphérique d'irradiance incidente à un point de la scène. Rappelons qu'elle contient, à chaque texel, la radiance entrante dans la direction correspondante. On peut utiliser cette information pour calculer une carte d'irradiance à partir de la carte de radiance ; il suffit d'appliquer l'intégrale à l'équation 2.1 pour chaque *texel*. L'intégrale pour le texel t_n se fera alors sur tous les texels correspondant à des directions dans l'hémisphère autour de t_n, dépendamment de la projection utilisée.

Les cartes d'irradiance sont couramment utilisées pour obtenir une évaluation rapide de l'irradiance de sources secondaires. On peut par exemple stocker l'irradiance provenant du ciel et éviter d'avoir à l'évaluer au moment du *shading*, ce qui peut nécessiter l'évaluation d'équations physiques complexes ou le stockage des facteurs nécessaires au calcul de ces équations [16].

Figure 2.7 – Exemple de carte environnementale HDR de *Grace Cathedral* [20].

2.8 *Irradiance volume*

Greger et Shirley [19] ont poussé l'idée d'utiliser les cartes d'environnement au niveau supérieur. S'il est possible de les utiliser pour stocker l'irradiance incidente à un point donné, il est également possible de les utiliser pour stocker et calculer une approximation de l'irradiance à tous les points au sein d'une scène.

Ils procèdent en construisant un *irradiance volume*, soit une structure qui contient un échantillonnage de la scène (sous la forme d'une carte environnementale) en plusieurs points. C'est à ces cartes que l'on réfère quand on parle de sondes d'irradiance (plus couramment appelés *irradiance probes* dans la littérature ; nous les désignerons ainsi dans le texte). Ils sont appelés comme tel puisqu'ils sont carrément un sondage de l'irradiance à un point donné.

On peut ensuite reconstruire une approximation de l'irradiance à un point arbitraire par interpolation. La fonction d'irradiance incidente est une fonction définie sur une sphère, qui varie généralement lentement pour des sources distantes de lumière ; l'interpolation entre des échantillons connus est donc une façon valide de la retrouver.

Greger et Shirley construisent leur volume en placant des échantillons sur une grille 3D espacée régulièrement. Si une des cellules de la grille contient de la géométrie, la cel-

lule est subdivisée en 8 cellules plus petites, et le processus est répété pour chaque sous-cellule ainsi créée jusqu'à une profondeur maximale. À chaque sommet des cellules, la fonction d'irradiance entrante est stockée pour un ensemble de directions découpant uniformément la sphère unitaire. La figure 2.8 montre le volume construit par les auteurs.

Cette structure permet d'obtenir rapidement une approximation de la contribution indirecte du *shading* pour n'importe quelle surface dynamique de la scène, à condition que les surfaces statiques composant l'environnement et les sources d'éclairage ne changent pas. Pour obtenir l'irradiance entrante à un point $\vec{p}$ ayant une normale $\vec{n}$, il suffit de trouver les échantillons entourant $\vec{p}$ et de faire la somme de la radiance vers $\vec{n}$ pour chacun de ces points, pondérés par les poids d'interpolation associés à chaque échantillon.

2.9 Interpolation

L'interpolation entre des échantillons est un outil de base des mathématiques ; il s'agit d'évaluer une fonction inconnue en se fiant sur des valeurs connues.

Soit une fonction $f(x)$, et un ensemble connu de valeurs de la fonction $f_0 = f(x_0), f_1 = f(x_1), ..., f_n = f(x_n)$. On peut écrire :

$$f(x) \approx \sum_N w_n f_n(x),$$

où les w_n sont les *poids d'interpolation*. Ceux-ci sont définis différemment selon la méthode d'interpolation employée.

Par exemple, l'*interpolation trilinéaire* requiert de placer les échantillons sur les points de maille d'une grille régulière (voir figure 2.9 pour la géométrie ; les C_{xyz} sont les échantillons de référence, et le point C est le point visé par l'interpolation). Pour interpoler un échantillon à la position $\vec{p} = (x, y, z)$, on trouve d'abord la cellule dans laquelle $\vec{p}$ se situe. On notera $\vec{p}_c = (x_c, y_c, z_c)$ la position de l'échantillon au sein de la cellule. Les poids seront alors (en supposant un espacement de 1 entre les points de maille) :

$$w_{abc} = X(a,x) \cdot Y(b,y) \cdot Z(c,z)$$

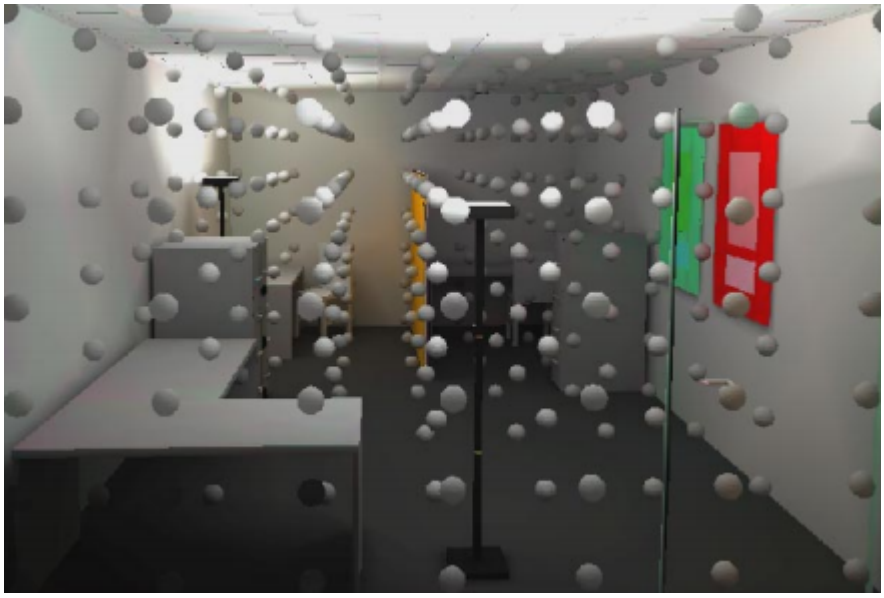


Figure 2.8 – Le volume d'irradiance présenté par Greger et Shirley [19].

avec

$$X(a,x) = \begin{cases} (1-x_c) & sia = 0 \\ x_c & sia = 1. \end{cases}$$

Les valeurs pour $Y(b,y)$ et $Z(c,z)$ se trouvent de la même façon [2].

D'autres méthodes ne requièrent pas de placer les points sur une grille régulière. Par exemple, une méthode simple est d'utiliser la *pondération inverse à la distance* (PID) [2] :

$$f(x) = \sum_N \frac{f_i(x_i)}{(x_i - x)^{\frac{\alpha}{2}}} \Bigg/ \sum_N \frac{1}{(x_i - x)^{\frac{\alpha}{2}}}. \tag{2.3}$$

La PID présente le défaut que chaque échantillon contribue au résultat final ; la valeur de l'exposant détermine l'importance relative des échantillons les plus près par rapport aux plus distants.

Dans le contexte des probes, une méthode intéressante est offerte par Cupisz [8], qui propose d'interpoler en formant une *tétraédralisation de Delauney* entre les probes. On obtient alors un maillage triangulaire, à l'intérieur duquel on peut interpoler selon les

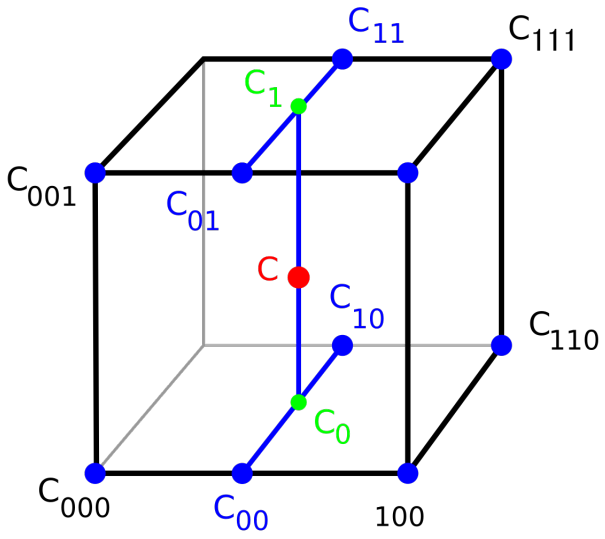


Figure 2.9 – Visualisation de la géométrie de l'interpolation trilinéaire [5].

coordonnées barycentriques du point par rapport au tétraèdre dans lequel il se trouve.

2.10 Harmoniques sphériques

La méthode décrite précédemment stocke la valeur de la fonction d'irradiance entrante à chaque point pour un ensemble de directions fixe. Il existe des méthodes plus efficaces pour la stocker ; notre travail emploie les *harmoniques sphériques* (SH), un outil mathématique couramment utilisé. Ses origines proviennent de la physique ; spécifiquement, elles sont la solution de l'équation de Laplace pour les variables angulaires [34]. Elles sont cruciales à la mécanique quantique, mais elles sont également utiles dans divers domaines, comme dans le cas qui nous intéresse.

Les harmoniques sphériques sont un ensemble de fonctions définies sur la sphère unitaire. Ce sont des fonctions complexes, mais on peut les exprimer sous une forme réelle [34] :

$$y_m^l = \begin{cases} \sqrt{2} K_l^m \cos(m\phi) P_l^m(\cos\theta), & m > 0 \\ \sqrt{2} K_l^m \sin(|m|\phi) P_l^{|m|}(\cos\theta), & m < 0 \\ K_l^0 P_l^0(\cos\theta), & m = 0, \end{cases}$$

où les (l,m) sont des indices dénotant la *bande* des coefficients, les K_l^m sont des constantes de normalisation, et les P_l^m sont les polynômes de Legendre. Les fonctions de base SH sont indexées selon la bande donnée par (l,m). La bande l contient $(2l+1)$ fonctions, indexées selon $m = -l, -l+1, ..., l-1, l$. Les 9 premières bandes sont représentées à la figure 2.10 : pour chaque direction sur les sphères, la longueur du vecteur pointant du centre de la sphère vers l'élément de surface correspondant à chaque direction est multiplié par la valeur absolue de l'évaluation de la fonction de base. Les valeurs positives sont colorées en rouge, les négatives en bleu. On a $l \in [0,\infty]$; il y a donc un nombre infini de fonctions de base SH.

La propriété clé de ces fonctions est qu'elles forment une base orthonormale ; elles nous permettent alors d'exprimer des fonctions sphériques avec un ensemble de coefficients. On retrouve un analogue des fonctions de Fourier, qui forment une base du même type et permettent d'exprimer des fonctions 2D cartésiennes comme une somme de fonctions trigonométriques pondérées. Les fonctions de Fourier se succèdent avec des fréquences de plus en plus hautes, et permettent ainsi de mesurer la réponse fréquentielle d'une fonction 2D. L'analogie est présente pour les SH ; leurs fréquences augmentent avec la bande, et leur orientation varie de façon à capturer les variations angulaires sur la fonction à représenter.

Formellement, on écrira :

$$f(\omega) = \sum_l \sum_{m=-l}^{l} f_l^m y_l^m(\omega)$$

où les y_l^m sont les valeurs des fonctions de base SH dans la direction ω et les f_l^m sont les coefficients pondérant chaque bande. Inversement, les f_l^m peuvent être générés par une opération similaire :

$$f_l^m = \int f(\omega)\, y_l^m(\omega) d\omega.$$

Il s'agit d'une opération de *projection*, où on fait passer la fonction de la base des coordonnées 3D à la base définie par les fonctions d'harmoniques sphériques.

C'est cette propriété que nous exploitons pour stocker l'irradiance entrante de nos

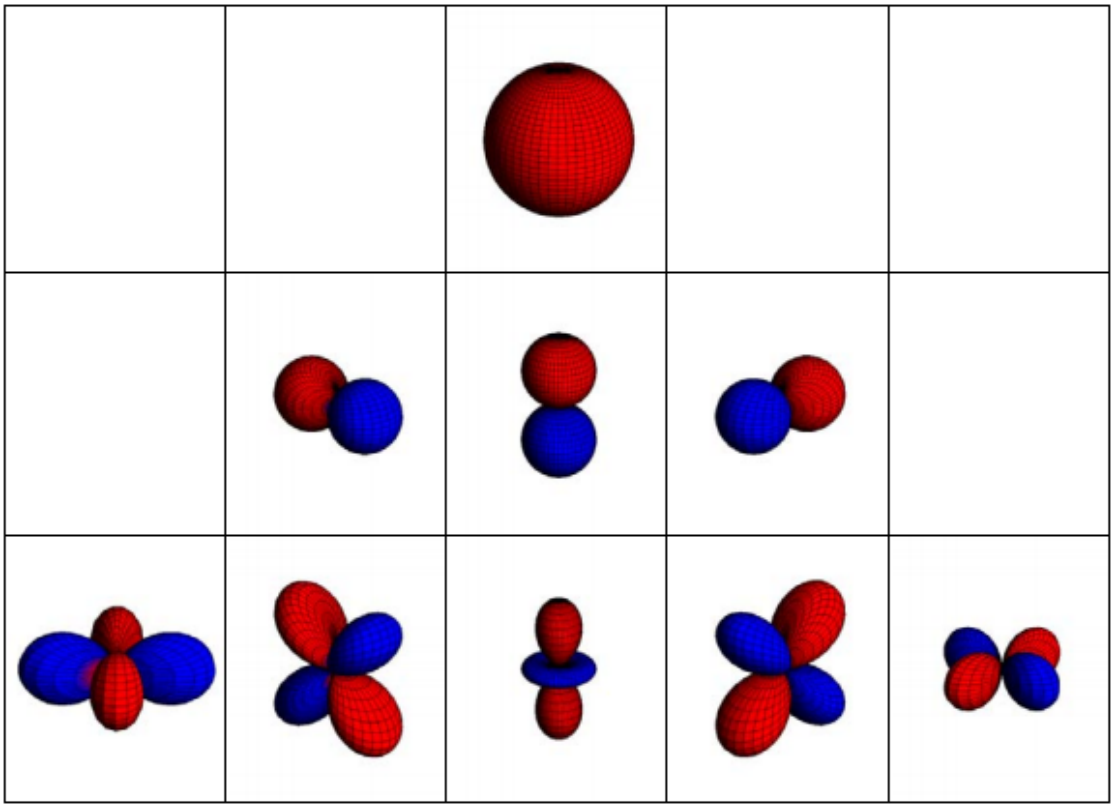


Figure 2.10 – Représentation des valeurs des 9 premières bandes SH [34].

probes de façon compacte, selon une méthode introduite par Ramamoorthi [31]. La projection de l'irradiance entrante sur les fonctions d'harmoniques sphériques pour un probe génère une série de coefficients E_l^m :

$$E_l^m = \int E(\omega) y_l^m(\omega) d\omega.$$

La même opération est effectuée non seulement pour la radiance, mais aussi pour le terme cosinus, et on peut donc exprimer l'irradiance entrante pour une normale $\vec{n}$ comme :

$$E(\vec{n}) = \sum_{l,m} A_l L_l^m y_l^m(\vec{n}).$$

L'astuce de Ramamoorthi est que les coefficients du cosinus clampé s'éteignent rapidement à mesure que l'on monte dans les bandes SH ; après $l = 2$, les valeurs deviennent négligeables :

$$\begin{aligned} A_0 &= 3.141593 \quad & A_1 &= 2.094395 \\ A_2 &= 0.785398 \quad & A_3 &= 0 \\ A_4 &= -0.13090 \quad & A_5 &= 0 \\ A_6 &= 0.049087 \quad & &\text{etc.} \end{aligned}$$

Ainsi, pour exprimer une approximation satisfaisante de l'irradiance incidente à un point, il suffit de stocker 9 coefficients d'harmoniques sphériques, la contribution des coefficients suivants étant minime grâce au terme cosinus. Cette expansion en coefficients d'harmoniques sphériques de l'irradiance est maintenant couramment utilisée dans les applications de rendu en temps réel, puisque les probes n'ont qu'à stocker 27 nombres à virgule flottante (9 par couleur). Plutôt que d'interpoler directement les valeurs d'irradiance, on peut interpoler les coefficients pour obtenir une version reconstruite de l'irradiance à n'importe quel point.

CHAPITRE 3

PROBLÈME

Dans ce chapitre, nous parlerons un peu plus en détail des probes d'irradiance ; nous décrirons notre problématique ainsi que le travail effectué auparavant en ce sens.

3.1 Placement des probes

La méthode des probes d'irradiance accélère le *shading* diffus en fournissant une approximation de la fonction d'irradiance incidente par interpolation entre des échantillons discrets. Bien que la description de la méthode soit simple, l'application requiert un soin particulier, puisque le résultat final dépend de paramètres qui sont uniques à chaque scène. Certains sont intrinsèques à la scène, tels que la géométrie et les sources lumineuses que l'on désire capturer. D'autres sont extrinsèques à la scène, tel que le budget mémoire alloué aux probes.

Greger et Shirley [19] ont placé leurs probes sur une grille régulière avec une subdivision hiérarchique, selon la présence ou l'absence de géométrie au sein des cellules. Ceci permet de reconstruire le *shading* par interpolation trilinéaire.

Ce type de placement offre l'avantage d'être plus ou moins automatique ; il suffit de donner une taille et une position à la grille, ainsi qu'une taille de cellule, et de laisser l'algorithme décider de subdiviser chaque cellule selon si elle contient ou ne contient pas de géométrie. Cependant, la géométrie de la scène peut être problématique pour l'algorithme ; une cellule en contenant beaucoup répartie uniformément pourrait se voir subdivisée un nombre immense de fois, menant à une explosion du nombre de cellules. Les subdivisions naïves en 3D font augmenter le nombre d'échantillons selon n^3. Ce taux de croissance est très rapide, il est donc important de limiter le nombre de divisions. On court alors le risque de ne pas capturer des détails qui auraient requis une grille plus fine pour être résolus.

Un autre problème est celui de la visibilité. Les probes ont beau être proprement

alignés sur les axes, rien ne garantit que dans le cas général la géométrie sera ainsi alignée, et un mur oblique passant à travers une structure droite introduira des artéfacts importants dans le *shading*. Un exemple d'un tel problème est visible à la figure 3.1 : des taches claires sont visibles autour de la porte. Ceci est dû au fait que la géométrie se trouve à l'intérieur d'une cellule dont les mailles ne sont pas toutes dans la pièce, et des probes situés à l'extérieur de la pièce participent à l'interpolation.

Ces facteurs montrent qu'on ne peut naïvement appliquer la méthode de Greger et Shirley en milieu industriel. Dans les jeux vidéo, les scènes sont souvent immenses et comportent une grande quantité de géométrie. De plus, les budgets consacrés à la mémoire et au temps de *shading* sont souvent très serrés, l'illumination indirecte étant seulement une partie de toutes les contributions menant à l'image finale. Si la structure de probes comporte trop d'échantillons, elle peut utiliser de grandes quantités d'espace dans le mémoire et sur le disque. De plus, il faut considérer le fait que plusieurs objets dynamiques à —emphshader implique plusieurs cellules à aller chercher en mémoire, ce qui peut poser des problèmes au niveau des accès à la mémoire. La question de l'optimisation des accès mémoire est au-delà de notre discussion, mais elle montre la portée des divers aspects à considérer lors de la création d'une structure de probes.

Ces problèmes peuvent être amoindris en travaillant soigneusement la géométrie dans la scène ainsi que les paramètres décrivant la structure de probes ; il faut cependant qu'un artiste consacre du temps et de l'énergie à ce processus. Le développement de scènes est long et complexe, et les spécifications des scènes créées changent souvent plusieurs fois en cours de route. Il faut alors que les artistes d'éclairage portent une attention constante à maintenir la validité de leurs probes.

Tel que mentionné précédemment, l'interpolation introduit nécessairement une erreur dans le résultat obtenu par rapport à la valeur analytique. Souvent, dans l'industrie, cette erreur n'est pas mesurée de façon objective. Les artistes doivent plutôt se fier sur leurs perceptions et tentent d'obtenir une reconstruction de l'irradiance qu'ils jugent correcte. Le processus inclut une grande part de subjectivité artistique, où le fait que le résultat soit plausible et agréable à l'oeil est jugé comme satisfaisant.

Le fait qu'une solution automatique pour le placement de probes dans une scène

Figure 3.1 – Exemple de *light leaking* dans le moteur de rendu de *Treyarch* [23].

soit désirable devient évident. Non seulement une telle méthode pourrait-elle permettre de libérer du précieux temps d'artiste ; elle pourrait également être formulée selon une perspective de *minimisation de l'erreur* de reconstruction du *shading*. Ceci pourrait nous garantir que les structures de probes produites seraient optimales, c'est-à-dire que le *shading* qu'elles produisent serait le plus près possible de la réponse attendue.

La minimisation d'erreur est un problème bien étudié, et d'exprimer notre problème comme tel nous donne accès à des algorithmes bien étudiés pour faciliter le processus.

3.2 Travail antérieur

Le placement automatique de probes est un domaine qui n'a pas été beaucoup étudié. En industrie, le placement des probes tend à être une arrière-pensée, et plus d'énergie est consacrée à utiliser les probes de façon plus sophistiquée que les volumes d'irradiance décrits précédemment.

Par exemple, dans *Far Cry 3* [16], l'outil de placement de probes d'*ambiance* (qui accumulent la lumière provenant du ciel et du soleil) lance un rayon vertical tous les 4 mètres et place un probe si le rayon touche de la géométrie. Ainsi, moins de probes

sont placés dans les grands espaces vides, tandis que les zones plus denses se verront allouer plus de probes. Les développeurs ont préféré plutôt consacrer leur énergie au *relighting* des probes, c'est-à-dire que leur contenu peut être modifié dynamiquement selon le temps de la journée. Cette méthode amène à un nombre immense de probes : jusqu'à 50 secteurs peuvent être chargés en mémoire à la fois autour du joueur, et chacun de ces secteurs contient en moyenne 70 probes. On peut voir qu'il y a alors matière à tenter de réduire ce nombre.

Dans *Killzone Shadow Fall* [37], les probes sont placés automatiquement sur les mailles d'une *voxelization* de la géométrie composant les niveaux ; des probes sont également ajoutés pour remplir les espaces vides. Bien que le processus soit automatique, il résulte en des quantités de probes immenses ; leur nombre atteint plusieurs centaines de milliers. On peut voir un exemple du résultat à la figure 3.2. Les probes blancs sont sur les surfaces, tandis que les probes en jaune sont les probes de remplissage d'espace vide. Ce type de sur-échantillonnage est précisément ce que nous tentons d'éviter, préférant nous concentrer sur la création de structures de probes dont l'optimalité peut être démontrée.

Chez *Treyarch* [23], les artistes d'éclairage construisent manuellement des volumes d'irradiance ajustés aux pièces présentes dans les scènes du jeu. Ceci a des avantages au niveau de la qualité ; par exemple, les fuites de lumière peuvent être minimisées. Cependant, le processus de construction manuelle des volumes introduit un coût de temps pour les artistes ; des outils ont donc dû être développés pour simplifier la construction des volumes. Un coût supplémentaire est introduit par l'obligation de maintenance des volumes au fil du développement ; des changements dans la géométrie des niveaux requièrent encore plus d'attention portée par les artistes aux volumes. Les auteurs notent avoir un algorithme de création automatique de volume offrant des résultats très satisfaisants dans certaines scènes, sans offrir de détails supplémentaires.

Le moteur de *Remedy* [33] utilise un volume d'irradiance sur une grille régulière pour l'illumination globale, en plus d'un système de probes pour les réflexions spéculaires. Ils stockent, dans chaque cellule du volume d'irradiance, de l'information supplémentaire sur la visibilité des probes de réflexions. Il est intéressant de noter que leurs probes

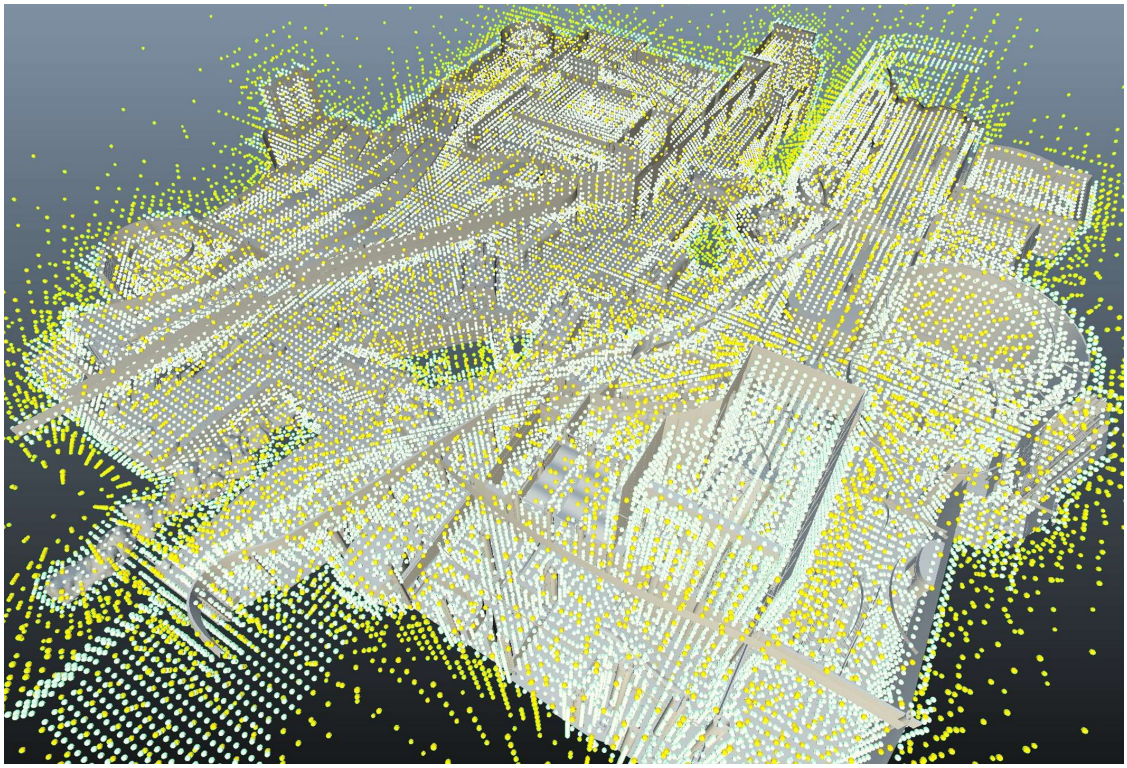


Figure 3.2 – Volumes de probes dans *Killzone Shadow Fall* [37].

de réflexions ne sont pas sur une grille (figure 3.3) ; ils sont placés automatiquement selon un algorithme de minimisation. La scène est suréchantillonnée à des emplacements réguliers, puis les K meilleurs probes sont choisis pour maximiser la surface de la scène visible par les probes, en minimisant la distance aux surfaces de la scène.

Au niveau académique, quelques chercheurs se sont penchés sur ce problème, mais il ne semble pas y avoir encore eu de percée majeure qui définit l'état de l'art. Dans sa thèse de maitrise *Automated Placement of Environment Probes* [11], Weis s'attaque au problème en utilisant une métrique d'erreur basée sur les différences entre les images produites par les probes.

Il génère une grande quantité de probes dans la scène, placés sur une grille régulière, puis définit, pour chacun de ces probes, un *vecteur d'intensité*. Celui-ci est exprimé comme

$$\vec{f} = \sum_{\omega_i \in \Omega} \omega_i \cdot \mathrm{lum}(E(\omega_i)),$$

où $E(\omega_i)$ est l'irradiance incidente encodée dans le probe pour la direction ω_i. Pour chaque texel sur l'image du probe, la direction correspondante est pondérée par la *luminance*. Cette quantité appartient à la *photométrie*. Ce domaine s'intéresse à la lumière telle que perçue par l'humain, à la différence de la radiométrie qui traite du transport de la lumière sans considérer sa perception. La luminance est la "brillance" d'une surface ; c'est un nombre dépendant de la somme de toutes les longueurs d'ondes présentes dans la lumière, pondérées par la réponse de l'oeil humain à chaque longueur d'onde.

Weis s'intéresse donc plus à comment les résultats seront perçus par les utilisateurs qu'au réel transport de lumière dans la scène. Il s'agit d'une perspective intéressante, la luminance étant une valeur très importante en *design* de couleurs [28].

Le vecteur d'intensité pointe dans la direction où, en moyenne, la luminance est la plus forte. Weis considère ces vecteurs de luminance pondérée comme une approximation du *champ d'irradiance incidente*. L'analogie avec le champ électrique est exploitée au maximum : Weis place ensuite des particules virtuelles à chaque point de maille dans le champ, et les laisse remonter le gradient du champ jusqu'à trouver les extrémums locaux.

Figure 3.3 – Probes de réflexions spéculaires dans le moteur de Remedy [33].

Les particules suivent la direction donnée par les vecteurs d'intensité, jusqu'à ce qu'elles sortent de la scène ou entrent dans des boucles fermées. Les premières sont jetées ; les positions des secondes sont alors considérées comme candidates pour le placement des probes finaux. L'intuition de Weis est que c'est à ces point qu'on veut capturer l'irradiance, puisque c'est à ces points que les variations du champ sont les plus importantes.

Weis rapporte que les particules tendent à former des grappes, c'est-à-dire qu'on peut en trouver plusieurs situées autour des points d'intérêt. Il est indésirable de placer plusieurs probes directement les uns à côté des autres ; l'auteur utilise donc une métrique de similarité entre les probes pour les fusionner en fonction de leur distance et de leur contenu. L'auteur utilise la *Earth Mover's Distance* [32], qui attribue une valeur de similarité à chaque paire de probes en fonction du travail à faire pour transformer un probe en un autre via la construction et l'analyse d'un histogramme de couleurs.

Weis obtient des résultats intéressants qui offrent un *shading* final plausible. Cependant, aucune métrique d'erreur objective n'est employée, et l'évaluation de ses résultats est faite de façon subjective.

Sukys, dans son mémoire *Light Probe Cloud Generation for Games* [35], examine

un algorithme de type *spawn and merge* pour le placement automatique. Il place une grande quantité de probes sur une grille régulière, puis examine, entre chaque voisin, la similarité de la projection en SH de l'irradiance. Si la similarité dépasse un critère de tolérance, les probes sont supprimés, et un nouveau probe est créé à leur point milieu. Le processus est répété jusqu'à ce qu'un critère soit rempli ; Sukys retient 3 critères, soit :

- le nombre de probes créés dépasse un maximum choisi,
- plus aucune paire de probes ne dépasse le seuil de tolérance de similarité, ou
- le nombre d'itérations dépasse un maximum choisi.

L'auteur note son insatisfaction face aux résultats offerts par sa méthode. Il note que les probes ont tendance à former des grappes très denses autour des sources de lumière. Un grand nombre de probes est potentiellement généré à chaque itération, selon le seuil de tolérance choisi. Il remarque également qu'il est difficile de faire l'équilibre entre une explosion du nombre de probes et un seuil de tolérance trop bas qui ratera des détails dans la fonction d'irradiance.

CHAPITRE 4

CONSTRUCTION AUTOMATIQUE D'ENSEMBLE DE PROBES OPTIMISÉS

4.1 Notre méthode

Les méthodes évaluées dans le chapitre précédent ont, selon nous, le défaut de ne pas évaluer de critère d'erreur objectif. Nous proposons donc une méthode pour placer des probes basée sur la minimisation d'une fonction d'erreur.

Plusieurs choix existent pour la définition de la fonction d'erreur. Nous avons choisi de travailler selon une approche standard en optimisation linéaire, soit avec les *moindres carrés*. Cette approche consiste à réduire la somme des différences au carré entre la valeur exacte et la valeur approximée de la fonction à des points du domaine.

On commence par définir un ensemble de probes *d'entraînement*. Ces probes constituent notre valeur de référence et représentent les valeurs cibles de la fonction à approximer. L'ensemble d'entraînement contiendra généralement un nombre de probes beaucoup plus important que le nombre de probes que nous souhaitons placer dans notre structure finale.

Soit une scène et ses valeurs de référence S et un ensemble de probes P fournissant une approximation de l'éclairage indirect pour S. Nous définissons $e(P,S)$ comme étant notre fonction d'erreur. En posant X comme l'ensemble des structures de probes qu'il est possible de construire dans la scène, nous voulons résoudre :

$$\underset{P \in X}{\text{minimiser}}\ e(P,S)$$

avec

$$e = \sum_{n=1}^{N} \sum_{l=0}^{2} \sum_{m=-l}^{l} (c_{n,l,m}^{\text{ref}} - c_{n,l,m}^{\text{calc}})^2 \tag{4.1}$$

où la somme se fait sur chaque coefficient de la projection SH de chaque probe de l'ensemble d'entraînement. Les $c_{n,l,m}^{\text{ref}}$ sont les coefficients SH des probes de référence, qui ne

changent pas, et les $c^{\text{calc}}_{n,l,m}$ sont les valeurs interpolées des coefficients selon la structure de probes en construction.

La valeur de la fonction d'erreur est considérée comme un pointage qui représente à quel point les valeurs calculées sont similaires aux valeurs de référence. Une valeur de 0 signifierait que nous arrivons parfaitement à représenter la fonction d'irradiance à tous les points d'échantillonnage de référence. Cependant, ceci ne garantirait pas que l'irradiance serait bien représentée *partout* dans la scène ; la répartition spatiale des échantillons de référence est importante. Nous plaçons nos échantillons à des points aléatoires uniformément répartis dans la scène, de façon à tenter de répartir l'erreur sur l'étendue maximale de la scène.

4.2 Variation des coefficients

Le déplacement d'un probe change non seulement l'information qu'il contient, mais également les poids associés à chaque probe lors de l'interpolation. La valeur de la fonction d'erreur sera donc modifiée directement par un moindre changement à la structure de probes. Notre méthode est d'abord de trouver les changements à la structure qui résultent en une fonction d'erreur plus petite, d'appliquer ces changements, puis de répéter l'opération jusqu'à ce que nous ne puissions plus trouver de changement qui mène à une diminution.

Il s'agit donc de trouver un ensemble de vecteurs $\Delta\vec{p}_n$ qui correspondent au déplacement appliqué au probe n. Pour trouver ces vecteurs, nous commençons par développer l'expression des valeurs des coefficients interpolés. Soit le coefficient dans la bande SH $s \in [0,8]$ pour la couleur $k \in [R,G,B]$ de l'échantillon p :

$$c_{psk}(\vec{p}_p, \vec{p}_0, ..., \vec{p}_n) = \sum_{n=1}^{N} w_n(\vec{p}_n)\, \tilde{c}_{nsk}(\vec{p}_n), \tag{4.2}$$

où la somme se fait sur les N probes. Les $\tilde{c}_{nsk}$ sont les coefficients des probes pour la bande SH et la couleur correspondante.

Si chaque probe n est déplacé par un déplacement $\Delta\vec{p}_n$, nous obtenons des nouveaux coefficients et des nouveaux poids d'interpolation :

$$c'_{psk}(\vec{p}_p, \vec{p}_0, ..., \vec{p}_N) = \sum_{n=0}^{N} w'_n(\vec{p} + \Delta\vec{p}_n)\, \tilde{c}'_{nsk}(\vec{p} + \Delta\vec{p}_n).$$

Pour calculer le changement $\Delta c_{psk} = (c'_{psk} - c_{psk})$ que subit chaque coefficient interpolé, il faut faire appel aux outils du calcul différentiel. c_{psk} est une fonction de plusieurs variables ; elle dépend de la position de l'échantillon $\vec{p}_p$, mais aussi de la position de tous les probes présents dans la structure. La figure 4.1 représente l'ajustement des poids suite au déplacement des probes.

La *différentielle totale* d'une fonction de plusieurs variables $f(x_0, ..., x_n)$ est donnée par la somme des dérivées de la fonction avec chacune de ses variables, pondérée par le changement dx_i appliqué à ces dernières :

$$df = \frac{\partial f}{\partial x_0} dx_0 + ... + \frac{\partial f}{\partial x_n} dx_n.$$

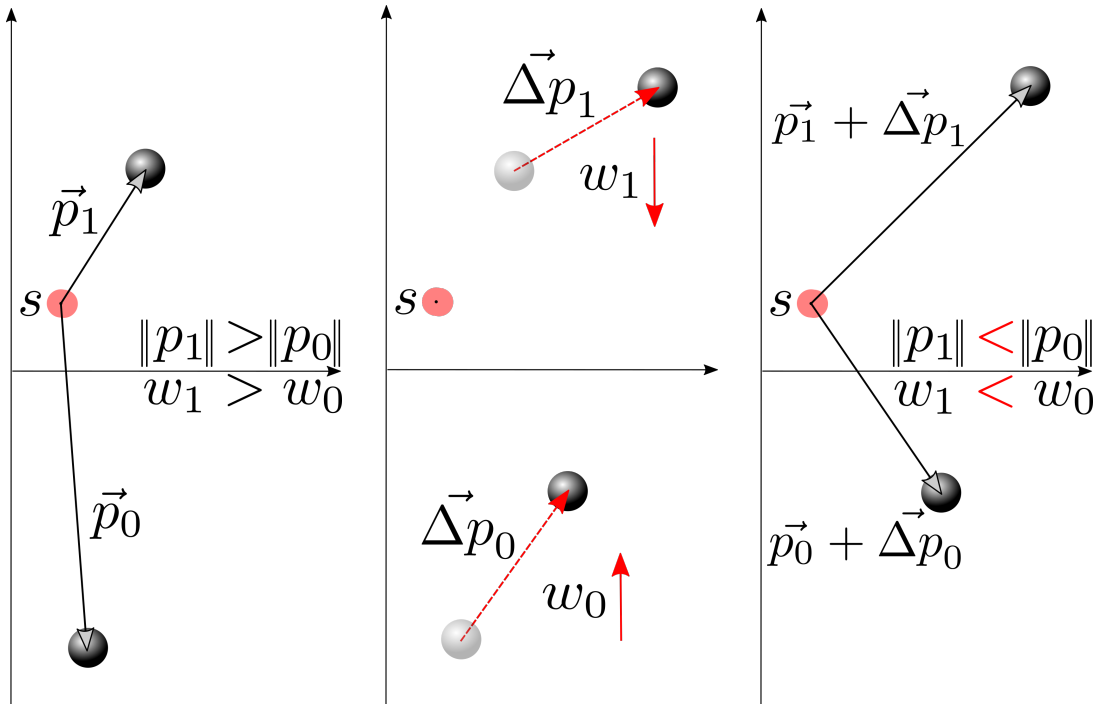
Dans le cas qui nous occupe, on cherche la différentielle totale de $c_{psk}(\vec{p}_p, \vec{p}_0, ..., \vec{p}_n)$ où $\vec{p}_p$ est la position de l'échantillon et $\vec{p}_n$ est la position du probe n. c_{psk} dépend aussi du contenu du probe n, mais puisque celui-ci est uniquement dépendant de la position du probe, il n'est pas inclus dans ses dépendances directes. On aura donc :

$$dc_{psk} = \frac{\partial c_{psk}}{\partial \vec{p}_p} d\vec{p}_p + \sum_{n=0}^{N} \frac{\partial c_{psk}}{\partial \vec{p}_n} d\vec{p}_n.$$

Puisque les échantillons sont fixes et qu'on s'intéresse plutôt à l'effet de déplacer les probes, le premier terme est toujours 0. La position étant une fonction des 3 dimensions de l'espace cartésien, on peut de la même façon séparer les termes de droite en 3 termes indépendants :

$$dc_{psk} = \sum_{n=0}^{N} \left(\frac{\partial c_{psk}}{\partial p_{nx}} dp_{nx} + \frac{\partial c_{psk}}{\partial p_{ny}} dp_{ny} + \frac{\partial c_{psk}}{\partial p_{nz}} dp_{nz} \right).$$

La variation d'un coefficient à un point d'échantillonnage dépend ainsi de l'action combinée de la variation de position de tous les probes dans la structure. L'évaluation des termes explicites de cette équation requiert un soin particulier ; nous y viendrons plus tard.



(a) Poids avant le déplacement des probes.

(b) Déplacement des probes et ajustement des poids.

(c) Poids après le déplacement des probes

Figure 4.1 – Représentation schématique de l'ajustement des poids d'interpolation suite à un déplacement de probes.

4.3 Formulation matricielle

Dans le cas qui nous occupe, nous connaissons la valeur des Δc_{psk} : ce sont les termes de notre fonction d'erreur (4.1). Nous les calculons à partir des valeurs connues des coefficients de l'ensemble d'entraînement et des valeurs approximées de l'ensemble d'interpolation.

$$\Delta c_{psk} = c_{psk} - \tilde{c}_{psk},$$

où c_{psk} est la valeur d'un coefficient pour le probe p dans l'ensemble d'entraînement, et $\tilde{c}_{psk}$ est la valeur interpolée à la position de p. Ces différences de coefficients Δc peuvent être très nombreuses ; on en trouve autant que le produit du nombre d'échantillons, du nombre de coefficients SH utilisés et de couleurs représentées dans le spectre lumineux.

La taille d'ensemble d'échantillons est un paramètre laissé à la discrétion de l'utilisateur lors de l'utilisation de l'algorithme. Le nombre d'échantillons optimal dépend de la taille de la scène et de la complexité des conditions d'éclairage à représenter. Nous avons utilisé 9 coefficients SH, conformément à ce que recommande Ramamoorthi, et 3 couleurs, l'espace RGB étant très standard.

Nous allons créer un vecteur $\vec{\Delta \mathbf{c}}$ contenant toutes les valeurs successives de Δc pour chaque probe de l'ensemble d'échantillonnage. Ce vecteur représente les variations que doivent subir chaque coefficient pour arriver à une fonction d'erreur de 0. Les valeurs qui changent de ligne en ligne sont montrées en rouge.

$$\vec{\Delta \mathbf{c}} = \begin{bmatrix} \Delta c_{s=0,(l,m)=(0,0),c=r} \\ \Delta c_{0,(0,0),g} \\ \Delta c_{0,(0,0),b} \\ \Delta c_{0,(1,-1),r} \\ \dots \\ \Delta c_{0,(2,2),b} \\ \Delta c_{1,(0,0),r} \\ \dots \\ \Delta c_{S,(2,2),b} \end{bmatrix}. \tag{4.3}$$

Nous allons ensuite construire un autre vecteur, $\vec{\Delta C}$, représentant la variation que nous voulons faire subir aux coefficients c_{psk}^{calc} interpolés par les probes de la structure en construction pour égaliser la variation ciblée dans $\vec{\Delta \mathbf{c}}$.

$$\vec{\Delta C} = \begin{bmatrix} \Delta c^{\text{calc}}_{0,(0,0),r} \\ \Delta c^{\text{calc}}_{0,(0,0),g} \\ \Delta c^{\text{calc}}_{0,(0,0),b} \\ \Delta c^{\text{calc}}_{0,(1,-1),r} \\ \dots \\ \Delta c^{\text{calc}}_{0,(2,2),b} \\ \Delta c^{\text{calc}}_{1,(0,0),r} \\ \dots \\ \Delta c^{\text{calc}}_{S,(2,2),b} \end{bmatrix} \tag{4.4}$$

$$= \begin{bmatrix} \sum_{n=0}^{N}(\frac{\partial c_{0,(0,0),r}}{\partial p_{nx}} dp_{nx} + \frac{\partial c_{0,(0,0),r}}{\partial p_{ny}} dp_{ny} + \frac{\partial c_{0,(0,0),r}}{\partial p_{nz}} dp_{nz}) \\ \sum_{n=0}^{N}(\frac{\partial c_{0,(0,0),g}}{\partial p_{nx}} dp_{nx} + \frac{\partial c_{0,(0,0),g}}{\partial p_{ny}} dp_{ny} + \frac{\partial c_{0,(0,0),g}}{\partial p_{nz}} dp_{nz}) \\ \sum_{n=0}^{N}(\frac{\partial c_{0,(0,0),b}}{\partial p_{nx}} dp_{nx} + \frac{\partial c_{0,(0,0),b}}{\partial p_{ny}} dp_{ny} + \frac{\partial c_{0,(0,0),b}}{\partial p_{nz}} dp_{nz}) \\ \sum_{n=0}^{N}(\frac{\partial c_{0,(1,-1),r}}{\partial p_{nx}} dp_{nx} + \frac{\partial c_{0,(1,-1),r}}{\partial p_{ny}} dp_{ny} + \frac{\partial c_{0,(1,-1),r}}{\partial p_{nz}} dp_{nz}) \\ \dots \\ \sum_{n=0}^{N}(\frac{\partial c_{0,(2,2),b}}{\partial p_{nx}} dp_{nx} + \frac{\partial c_{0,(2,2),b}}{\partial p_{ny}} dp_{ny} + \frac{\partial c_{0,(2,2),b}}{\partial p_{nz}} dp_{nz}) \\ \sum_{n=0}^{N}(\frac{\partial c_{1,(0,0),r),b}}{\partial p_{nx}} dp_{nx} + \frac{\partial c_{1,(0,0),r}}{\partial p_{ny}} dp_{ny} + \frac{\partial c_{1,(0,0),r}}{\partial p_{nz}} dp_{nz}) \\ \dots \\ \sum_{n=0}^{N}(\frac{\partial c_{S,(2,2),b}}{\partial p_{nx}} dp_{nx} + \frac{\partial c_{S,(2,2),b}}{\partial p_{ny}} dp_{ny} + \frac{\partial c_{S,(2,2),b}}{\partial p_{nz}} dp_{nz}) \end{bmatrix}. \quad (4.5)$$

Nous allons considérer que les déplacements de probes $\Delta\vec{p}$ sont si petits que nous pouvons poser $\Delta c_{psk}^{\text{calc}} = dc_{psk}^{\text{calc}}$. C'est ce qui nous permet de poser l'équation (4.5) comme une égalité. Chaque élément du vecteur est une longue somme de termes dépendant de deux facteurs. On peut réécrire $\vec{\Delta C}$ comme le produit de deux matrices : une matrice $\vec{\Delta P}$ contenant les déplacements $\Delta\vec{p}_{pi}$ (avec $i \in (x,y,z)$), et une matrice M contenant les $\frac{\partial c_{0,(0,0),r}}{\partial p_{nx}}$. On aura donc :

$$\vec{\Delta C} = M \cdot \vec{\Delta P}$$

avec :

$$\vec{\Delta P} = \begin{bmatrix} \Delta\vec{p}_{0x} \\ \Delta\vec{p}_{0y} \\ \Delta\vec{p}_{0z} \\ \Delta\vec{p}_{1x} \\ \dots \\ \Delta\vec{p}_{Nz} \end{bmatrix} \quad (4.6)$$

et

$$M = \begin{pmatrix} \frac{\partial c_{0,(0,0),r}}{\partial p_{0x}} & \frac{\partial c_{0,(0,0),r}}{\partial p_{0y}} & \frac{\partial c_{0,(0,0),r}}{\partial p_{0z}} & \frac{\partial c_{0,(0,0),r}}{\partial p_{1x}} & \cdots & \frac{\partial c_{0,(0,0),r}}{\partial p_{Nz}} \\ \frac{\partial c_{0,(0,0),g}}{\partial p_{0x}} & \frac{\partial c_{0,(0,0),g}}{\partial p_{0y}} & \frac{\partial c_{0,(0,0),g}}{\partial p_{0z}} & \frac{\partial c_{0,(0,0),g}}{\partial p_{1x}} & \cdots & \frac{\partial c_{0,(0,0),g}}{\partial p_{Nz}} \\ \frac{\partial c_{0,(0,0),b}}{\partial p_{0x}} & \frac{\partial c_{0,(0,0),b}}{\partial p_{0y}} & \frac{\partial c_{0,(0,0),b}}{\partial p_{0z}} & \frac{\partial c_{0,(0,0),b}}{\partial p_{1x}} & \cdots & \frac{\partial c_{0,(0,0),b}}{\partial p_{Nz}} \\ \frac{\partial c_{0,(1,-1),r}}{\partial p_{0x}} & \frac{\partial c_{0,(1,-1),r}}{\partial p_{0y}} & \frac{\partial c_{0,(1,-1),r}}{\partial p_{0z}} & \frac{\partial c_{0,(1,-1),r}}{\partial p_{1x}} & \cdots & \frac{\partial c_{0,(1,-1),r}}{\partial p_{Nz}} \\ \frac{\partial c_{0,(1,-1),g}}{\partial p_{0x}} & \frac{\partial c_{0,(1,-1),g}}{\partial p_{0y}} & \frac{\partial c_{0,(1,-1),g}}{\partial p_{0z}} & \frac{\partial c_{0,(1,-1),g}}{\partial p_{1x}} & \cdots & \frac{\partial c_{0,(1,-1),g}}{\partial p_{Nz}} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial c_{0,(2,2),b}}{\partial p_{0x}} & \frac{\partial c_{0,(2,2),b}}{\partial p_{0y}} & \frac{\partial c_{0,(2,2),b}}{\partial p_{0z}} & \frac{\partial c_{0,(2,2),b}}{\partial p_{1x}} & \cdots & \frac{\partial c_{0,(2,2),b}}{\partial p_{Nz}} \\ \frac{\partial c_{1,(0,0),r}}{\partial p_{0x}} & \frac{\partial c_{1,(0,0),r}}{\partial p_{0y}} & \frac{\partial c_{1,(0,0),r}}{\partial p_{0z}} & \frac{\partial c_{1,(0,0),r}}{\partial p_{1x}} & \cdots & \frac{\partial c_{1,(0,0),r}}{\partial p_{Nz}} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial c_{S,(2,2),b}}{\partial p_{0x}} & \frac{\partial c_{S,(2,2),b}}{\partial p_{0y}} & \frac{\partial c_{S,(2,2),b}}{\partial p_{0z}} & \frac{\partial c_{S,(2,2),b}}{\partial p_{1x}} & \cdots & \frac{\partial c_{S,(2,2),b}}{\partial p_{Nz}} \end{pmatrix}. \quad (4.7)$$

La matrice M semble complexe à première vue, mais son contenu est relativement simple. Chaque ligne représente une couleur différente du spectre RGB. À chaque fois que le cycle de couleurs recommence, l'indice de bande SH est augmenté. Enfin, quand toutes les bandes SH et les couleurs du premier échantillon ont été passées, on passe à l'échantillon suivant, jusqu'à atteindre l'échantillon N. On y trouve donc $9 \times 3 \times N$ lignes.

Les colonnes de M passent successivement les dérivées d'un coefficient par rapport aux 3 axes, et ce pour chaque probe présent dans la structure actuelle. Les colonnes sont au nombres de $3 \times P$, semblable à la taille du vecteur $\vec{\Delta P}$.

Cette façon d'exprimer le problème nous permet de procéder à l'envers. Nous savons que nous voulons que $\vec{\Delta C} = \vec{\Delta \mathbf{c}}$. De plus, nous pouvons exprimer $\vec{\Delta C}$ comme la multiplication d'une matrice M dont les éléments sont connus et d'un vecteur $\vec{\Delta P}$. Ce sont les éléments de ce dernier que nous chercherons. Nous sommes presque prêts à expliquer notre processus itératif ; il importe cependant, d'abord, d'obtenir la valeur exacte des termes contenus dans M.

4.4 Calcul explicite de la variation des coefficients

En utilisant l'équation 4.2, on peut écrire la valeur de chacune de ces dérivées ainsi :

$$\frac{\partial c_{psk}}{\partial p_{nx}} = \frac{\partial}{\partial p_{nx}} \sum_{m=1}^{N} w_m(\vec{p}) c_{mk}(\vec{p_m}).$$

La dérivée étant un opérateur linéaire, elle s'applique à tous les termes dans la somme :

$$\frac{\partial c_{psk}}{\partial p_{nx}} = \sum_{m=1}^{N} \frac{\partial}{\partial p_{nx}} \left(w_m(\vec{p}) c_{mk}(\vec{p_m}) \right).$$

Chacun des termes dans les dérivées est un produit de deux fonctions qui dépendent de p_{nx}. On peut appliquer la règle de chaîne, qui établit que pour une fonction f qui est le produit de deux fonctions $A(x)$ et $B(x)$, on a :

$$f(x) = A(x)B(x)$$

$$\frac{df}{dx} = \frac{dA(x)}{dx} B(x) + A(x) \frac{dB(x)}{dx}.$$

Dans le cas qui nous occupe, on a $A = w_m(\vec{p})$ et $B = c_{mk}(\vec{p_m})$. Ainsi :

$$\frac{\partial c_{psk}}{\partial p_{nx}} = \sum_{m=1}^{N} \left(\frac{\partial w_m(\vec{p})}{\partial p_{nx}} c_{mk}(\vec{p_m}) + w_m(\vec{p}) \frac{\partial c_{mk}(\vec{p_m})}{\partial p_{nx}} \right).$$

Pour trouver la variation associée à chaque coefficient pour le déplacement d'un probe, il nous faut trouver la variation des coefficients SH pour le déplacement de ce probe, ainsi que la variation des poids d'interpolation pour ce même déplacement.

4.5 Gradients de coefficients SH

Les gradients de coefficients SH ont été introduits par Annen [13], qui les a utilisés pour améliorer le *shading* produit par une source lumineuse située à une distance intermédiaire. En général, les coefficients pour un objet dynamique ne sont pas interpolés

par sommet, ce qui serait trop coûteux, les objets représentés dans les scènes de nos jours comportant des dizaines de milliers de triangles. Les coefficients interpolés sont plutôt calculés au centroïde de l'objet.

Annen utilise le gradient des coefficients SH pour approximer la variation des coefficients SH aux sommets de l'objet à *shader*. Les sommets étant plus près du centre de l'objet par rapport aux sources lumineuses, l'expansion de Taylor au premier degré est suffisante pour obtenir une bonne approximation. Le calcul peut rapidement être fait dans un *vertex shader* ; il suffit de passer le gradient des coefficients et la position du centroïde. La figure 4.2 montre l'effet d'utiliser ou pas les gradients des coefficients avec un probe situé entre les surfaces émissives ; on voit que l'objet dans la rangée du bas a un éclairage beaucoup plus crédible.

Annen calcule les gradients de façon analytique ; nous avons préféré, par simplicité d'implémentation, les calculer par échantillonnage du voisinage autour de chaque probe. Lors du rendu d'un probe, on crée six probes temporaires supplémentaires. Ces probes sont déplacés d'une petite distance positive et négative dans les trois axes. Le *théorème des différences finies*, schématisé à la figure 4.3, nous permet d'obtenir une estimation du gradient pour un coefficient c :

$$\nabla(c)_x = \frac{c_{x+d} - c_{x-d}}{d},$$

où c_{x+d} est le coefficient correspondant pour le probe temporaire à $x+d$ et similairement pour le second terme.

Cette méthode est simple à exécuter, mais comporte des désavantages : le gradient obtenu n'est qu'une estimation du gradient réel. Nous acceptons cette estimation, puisque nous sommes surtout intéressés à trouver la tendance générale de variation des coefficients SH. Nous recalculons le gradient à chaque itération, et bien que nous risquons d'avoir des itérations de trop pour corriger les erreurs dues à l'estimation du gradient, la convergence finale ne devrait pas être affectée. Tant que nous gardons les déplacements petits pour les probes temporaires, le gradient obtenu devrait être une assez bonne estimation du gradient analytique.

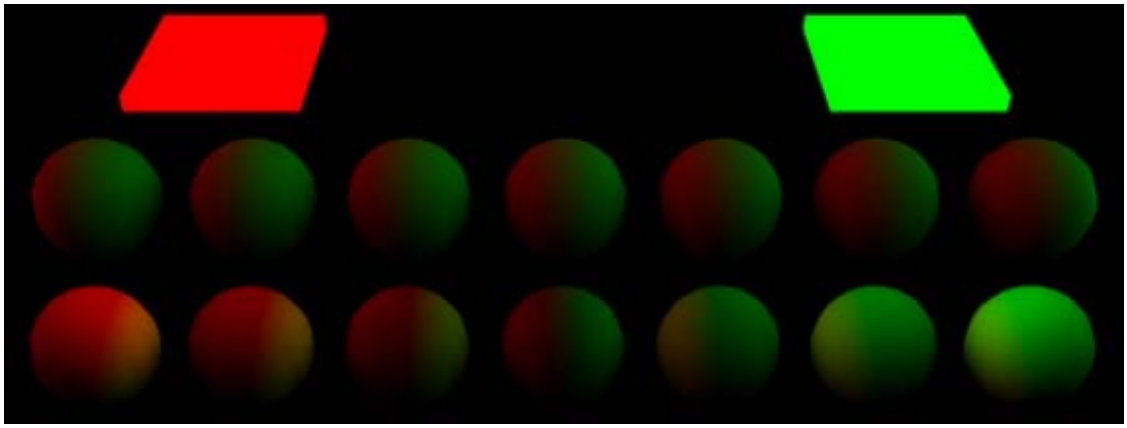


Figure 4.2 – Comparaison de l'utilisation (bas) ou pas (haut) des gradients SH pour le rendu [36].

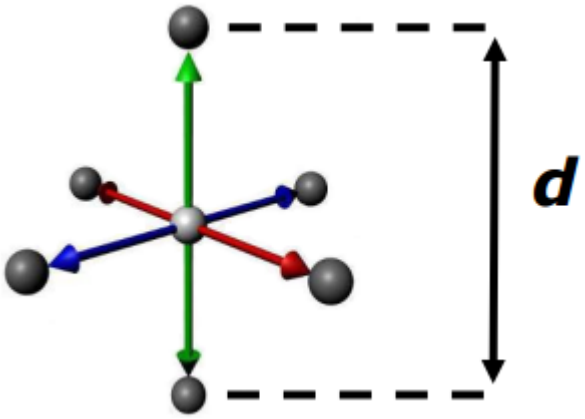


Figure 4.3 – Positions des probes temporaires (gris) pour l'évaluation du gradient d'un probe (blanc) [36].

4.6 Gradients des poids d'interpolation

Les poids d'interpolation sont délicats à sélectionner ; ils découlent directement du choix de méthode d'interpolation. L'interpolation trilinéaire et l'interpolation par coordonnées barycentriques d'un maillage de probes sont les choix les plus attrayants en termes de flexibilité et de qualité du rendu final. Cependant, ils présentent tous deux un inconvénient majeur : le poids d'interpolation $w(p_s, p_p)$ pour le probe p et l'échantillon s est une fonction discontinue sur le domaine pour ces deux méthodes. Ceci signifie que nous ne pouvons pas appliquer notre méthode : la construction de la matrice de poids d'interpolation exige de calculer la dérivée, et donc de choisir une fonction continue en tout point du domaine de dérivation.

Nous avons décidé d'utiliser la *pondération inverse à la distance* (PID) (équation 2.3) ; cette fonction a l'avantage d'être continue partout ; voyons comment sa dérivée est définie.

Les termes de la matrice M sont les dérivées par rapport au mouvement d'un probe arbitraire. Il nous faut donc examiner la dérivée $w_m(\vec{p})$ par rapport au déplacement d'un probe n. De plus, chaque terme de la matrice réfère au mouvement du probe dans un axe particulier ; nous ferons ici le développement pour l'axe x mais il est applicable de la même façon pour tous les axes.

$$\frac{\partial w_m(\vec{p})}{\partial p_{nx}} = \frac{\partial}{\partial p_{nx}} \left(\frac{1}{\sum_{i=0}^{N} \frac{1}{\|\vec{p}-\vec{p}_i\|^\alpha}} \frac{1}{\|\vec{p}-\vec{p_m}\|^\alpha} \right).$$

Nous sommes encore une fois en présence du produit de deux fonctions contenant p_{nx}, et la règle de chaîne s'applique :

$$\frac{\partial w_m(\vec{p})}{\partial p_{nx}} = \frac{\partial}{\partial p_{nx}} \left(\frac{1}{\sum_{i=0}^{N} \frac{1}{\|\vec{p}-\vec{p}_i\|^\alpha}} \right) \frac{1}{\|\vec{p}-\vec{p_m}\|^\alpha} + \frac{1}{\sum_{i=0}^{N} \frac{1}{\|\vec{p}-\vec{p}_i\|^\alpha}} \frac{\partial}{\partial p_{nx}} \left(\frac{1}{\|\vec{p}-\vec{p_m}\|^\alpha} \right).$$

Nous devons trouver deux dérivées ; commençons par la première :

$$\frac{\partial}{\partial p_{nx}}\left(\frac{1}{\sum_{i=0}^{N}\frac{1}{\|\vec{p}-\vec{p}_i\|^{\alpha}}}\right) = \frac{\partial}{\partial p_{nx}}\left(\sum_{i=0}^{N}\frac{1}{\|\vec{p}-\vec{p}_i\|^{\alpha}}\right)^{-1}$$

$$= \frac{\partial}{\partial p_{nx}}\left(\sum_{i=0}^{N}\|\vec{p}-\vec{p}_i\|^{-\alpha}\right)^{-1}$$

$$= -1\left(\sum_{i=0}^{N}\|\vec{p}-\vec{p}_i\|^{-\alpha}\right)^{-\alpha}\frac{\partial}{\partial p_{nx}}\left(\sum_{i=0}^{N}\|\vec{p}-\vec{p}_i\|^{-\alpha}\right). \tag{4.8}$$

La somme à l'intérieur de la dérivée se fait sur les positions de tous les probes ; nous pouvons distribuer la dérivée à l'intérieur de la somme, ce qui donne :

$$\frac{\partial}{\partial p_{nx}}\left(\sum_{i=0}^{N}\|\vec{p}-\vec{p}_i\|^{-\alpha}\right) = \frac{\partial}{\partial p_{nx}}\|\vec{p}-\vec{p}_0\|^{-\alpha} + \ldots + \frac{\partial}{\partial p_{nx}}\|\vec{p}-\vec{p}_P\|^{-\alpha}.$$

Le seul terme dans la somme qui ne soit pas égal à 0 est celui où $i = n$; écrivons le explicitement :

$$\frac{\partial}{\partial p_{nx}}\left(\sum_{i=0}^{N}\|\vec{p}-\vec{p}_i\|^{-\alpha}\right) = \frac{\partial}{\partial p_{nx}}\left(\|\vec{p}-\vec{p}_n\|^{-\alpha}\right).$$

Nous pouvons appliquer la dérivée, et du même coup ramener les termes ci-haut pour obtenir :

$$\frac{\partial}{\partial p_{nx}}\left(\frac{1}{\sum_{i=0}^{N}\frac{1}{\|\vec{p}-\vec{p}_i\|^{\alpha}}}\right) = -\alpha\|\vec{p}-\vec{p}_n\|^{-(\alpha+1)}\frac{\partial\|\vec{p}-\vec{p}_n\|}{\partial p_{nx}}$$

$$= \frac{-\alpha}{2\|\vec{p}-\vec{p}_n\|^{-(\alpha+2)}}\frac{\partial\left((p_x-p_{nx})^2+(p_y-p_{ny})^2+(p_z-p_{nz})^2\right)}{\partial p_{nx}}. \tag{4.9}$$

Encore une fois, nous pouvons distribuer la dérivée dans la somme. À partir d'ici, le résultat de la dérivée dépend de l'axe par rapport à lequel nous dérivons. La transposition à un autre axe est directe : il suffit de remplacer x par l'axe approprié. Les dérivées ne correspondant pas à l'axe de dérivation seront toujours 0 :

$$\frac{\partial\big((p_x-p_{nx})^2\big)}{\partial p_{nx}} = 2(p_x-p_{nx})\frac{\partial\big((p_x-p_{nx})\big)}{\partial p_{nx}} = -2(p_x-p_{nx}).$$

On peut enfin rassembler le tout :

$$\frac{\partial}{\partial p_{nx}}\left(\frac{1}{\sum_{i=0}^{N}\frac{1}{\|\vec{p}-\vec{p}_i\|^\alpha}}\right) = \frac{-\alpha}{2\|\vec{p}-\vec{p}_n\|^{-(\alpha+2)}}(-2(p_x-p_{nx}))$$

$$= \frac{\alpha(p_x-p_{nx})}{\|\vec{p}-\vec{p}_n\|^{(\alpha+2)}}.$$

Il nous manque seulement à trouver la seconde dérivée dans 4.6 ; cependant, nous la connaissons déjà ! Elle figure parmi les étapes que nous avons développées précédemment :

$$\frac{\partial}{\partial p_{nx}}\left(\frac{1}{\|\vec{p}-\vec{p}_m\|^\alpha}\right) = \begin{cases} 0 & m \neq n \\ \frac{-\alpha(p_x-p_{nx})}{\|\vec{p}-\vec{p}_n\|^{(\alpha+1)}} & m = n \end{cases}. \tag{4.10}$$

Une fois le développement complété, on peut enfin écrire la dérivée des poids d'interpolation pour la PID :

$$\frac{\partial w_m(\vec{p})}{\partial p_{nx}} = \frac{\alpha(p_x-p_{nx})}{\|\vec{p}-\vec{p}_n\|^{(\alpha+2)}}\frac{1}{\|\vec{p}-\vec{p}_m\|^\alpha} + \frac{1}{\sum_{i=0}^{N}\frac{1}{\|\vec{p}-\vec{p}_i\|^\alpha}}\frac{\partial}{\partial p_{nx}}\left(\frac{1}{\|\vec{p}-\vec{p}_m\|^\alpha}\right)$$

où le dernier facteur est donné par l'équation (4.10).

4.7 Processus d'optimisation et implémentation

Nous sommes en présence d'une équation linéaire de type

$$M\cdot\vec{x} = \vec{b}.$$

Ce type d'équation est bien connu, et il existe de nombreuses méthodes pour le résoudre. Nous utiliserons la méthode du *gradient conjugué*, implémentée dans la li-

brairie *Eigen* [12]. Le gradient conjugué requiert que M soit symétrique. Dans notre cas, la matrice ne sera symétrique que dans le cas particulier où $P = \frac{N}{9}$. Nous ne pouvons donc pas garantir la symétrie, et appliquons alors une transformation à l'équation en la prémultipliant par M^{T}. Nous résoudrons donc plutôt l'équation

$$M^{\mathrm{T}}M \cdot \vec{x} = M^{\mathrm{T}} \cdot \vec{b}.$$

$M^{\mathrm{T}}M$ étant symétrique par définition, nous pouvons utiliser le gradient conjugué. Puisque M^{T} est multiplié des deux côtés de l'équation, le résultat obtenu ne changera pas.

Nous pouvons maintenant définir le processus itératif que nous appliquerons pour chercher la meilleure valeur de la fonction d'erreur. Il s'agit de construire peu à peu un ensemble de probes d'interpolation qui auront la fonction d'erreur la plus petite possible. Le processus se fait en deux phases.

La phase 1 consiste en la recherche d'un emplacement optimal pour ajouter un probe à la structure existante. Un nombre choisi par l'utilisateur de points dans le domaine sont sélectionnés aléatoirement et un probe temporaire est placé tour à tour à chacun d'entre eux. Les points situés à l'intérieur de la géométrie de la scène sont éliminés, puisqu'ils ne peuvent contribuer de façon juste à l'évaluation de la fonction d'irradiance. Le probe temporaire est intégré à la structure existante, et la fonction d'erreur est évaluée en appliquant l'équation 4.1. Les coefficients SH sont interpolés à chaque point d'échantillonnage et chacun est soustrait à la valeur de référence contenue dans l'échantillon. Les résidus obtenus sont mis au carré puis sommés, ce qui donne l'évaluation de l'erreur globale.

Le probe ayant offert la meilleure performance, c'est-à-dire celui dont l'ajout diminue au maximum la fonction d'erreur, est gardé et intégré à la structure de façon permanente.

La seconde phase du processus consiste, de façon itérative, à trouver une série de déplacements de probes qui contribuent à faire descendre encore plus la fonction d'erreur. Les matrices et vecteurs présentés ci-haut sont calculés, puis l'algorithme du gra-

dient conjugué est appliqué. Celui-ci produit un vecteur $\vec{x}$ qui, multiplié à M, donne le résultat le plus près possible de $\vec{b}$.

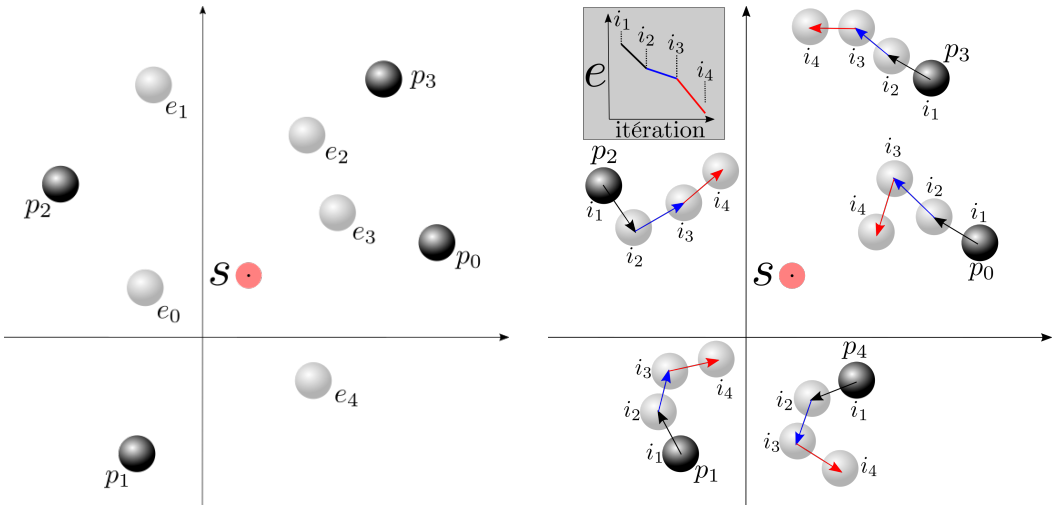
Le vecteur $\vec{x}$ contient les vecteurs de déplacements à appliquer à chaque probe. Le gradient conjugué tentant d'obtenir une valeur exacte de $\vec{b}$, les vecteurs obtenus sont généralement très longs. Ils ne peuvent être utilisés ainsi, puisque l'utilisation de l'approximation en série de Taylor pour les nouveaux coefficients requiert que les déplacements soient gardés petits. Les vecteurs sont donc normalisés puis divisés par un facteur approprié afin que leur longueur ne dépasse pas un seuil choisi par l'utilisateur en fonction des dimensions du domaine et de la fréquence de variation de la fonction d'irradiance.

La figure 4.4 montre les deux phases de notre processus à un haut niveau. Dans la première, des positions sont testées (représentées par les sphères pales) et provoquent des erreurs e_n ; un probe est placé à la position donnant la meilleure erreur (p_4 sur le schéma). Dans la phase 2, les probes se déplacent pendant 4 itérations $i_1, ..., i_4$ dans les directions faisant descendre la fonction d'erreur.

Les vecteurs de déplacement doivent également être analysés afin de s'assurer que le probe ne traversera pas une surface de la scène. Une structure d'accélération permet de tester si deux points sont situés de chaque côté d'un triangle composant la scène. Si c'est le cas, le vecteur est ajusté de façon à ne garder que la composante du vecteur parallèle à la surface, de façon à ce que le probe "glisse" sur la surface (figure 4.5).

Les probes déplacés doivent être regénérés ; le rendu des nouveaux probes peut prendre un temps considérable, dépendamment de la qualité choisie. Une fois les rendus terminés, la fonction d'erreur est de nouveau évaluée. Si celle-ci est plus basse qu'à l'opération précédente, la phase 2 est appliquée de nouveau à la structure de probes courante. Autrement, l'utilisateur peut choisir un nombre d'échecs successifs à endurer avant que l'algorithme ne revienne à la phase 1.

Nous appliquons donc un processus de stress suivi d'une relaxation ; l'optimisation successive avec petits pas fait que nous trouvons des *puits* locaux de l'erreur associée à notre fonction d'irradiance. Quand un nouveau probe est ajouté, l'état d'optimalité est perturbé, et les puits se déplacent. Nous croyons que la chaîne d'ajout-relaxation, com-



(a) Représentation schématique, phase 1. (b) Représentation schématique, phase 2.

Figure 4.4 – Représentation schématique des deux phases de notre optimisation.

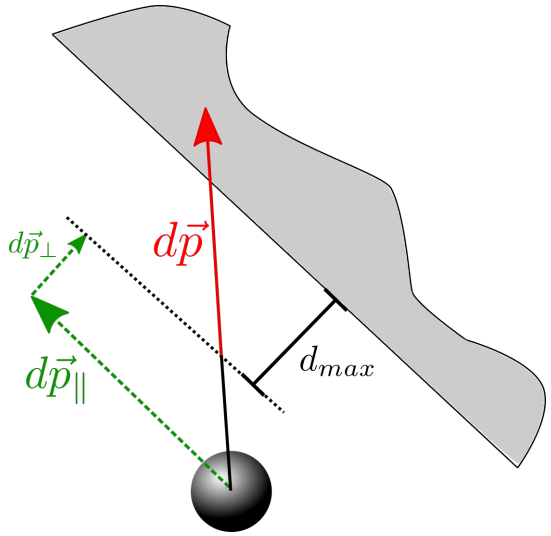


Figure 4.5 – Séparation du déplacement de probe en composantes parallèles et perpendiculaires dans le cas où la cible est trop près d'une surface.

binée au fait que nous évaluons un grand nombre d'emplacements potentiels pour les ajouts, ce qui nous donne une vue d'ensemble de l'impact de l'ajout d'un probe, contribue à créer une structure de probes optimale pour l'ensemble d'échantillons considéré.

CHAPITRE 5

RÉSULTATS

5.1 Validité des approximations

La première étape, avant de tester la construction de structures de probes au sein d'une vraie scène, est de vérifier si la phase 2 de la méthode fonctionne. Nous avons supposé qu'il était possible d'utiliser le gradient conjugué pour optimiser directement les coefficients SH, et bien que nous ayons tenté d'être rigoureux dans notre traitement mathématique, nous ne savons pas comment notre algorithme va vraiment se comporter avant de le tester.

Nous avons donc commencé par travailler avec un seul échantillon d'entraînement et un seul probe ; si notre processus fonctionne, l'optimisation devrait pouvoir "guider" le probe vers la position de l'échantillon. Si cette étape fonctionne, l'idée de construire une structure de probes à partir de rien pour plusieurs échantillons deviendrait réalisable. Nous avons utilisé la boite de Cornell [4] comme notre première scène.

La figure 5.1 montre la composante directe de l'illumination ; cette composante est simple et rapide à calculer (de façon approximative) sur une carte graphique. Les ombres sont complètement dures, et les surfaces dans les zones d'ombre sont complètement obscurcies. Il n'y a non plus aucune interréflexion entre les surfaces.

5.1.1 Test de la phase 2

Nous appliquons notre premier test en plaçant un échantillon dans la scène, puis en plaçant un probe à un point aléatoire. On peut observer le *shading* produit par ce probe à la figure 5.2.

Le probe étant très près du mur rouge, la composante rouge de ses coefficients SH a une importance démesurée par rapport aux couleurs réelles. La fonction d'erreur, pour cette configuration, a une valeur de 1.224. L'application d'une itération du processus d'optimisation produit un vecteur :

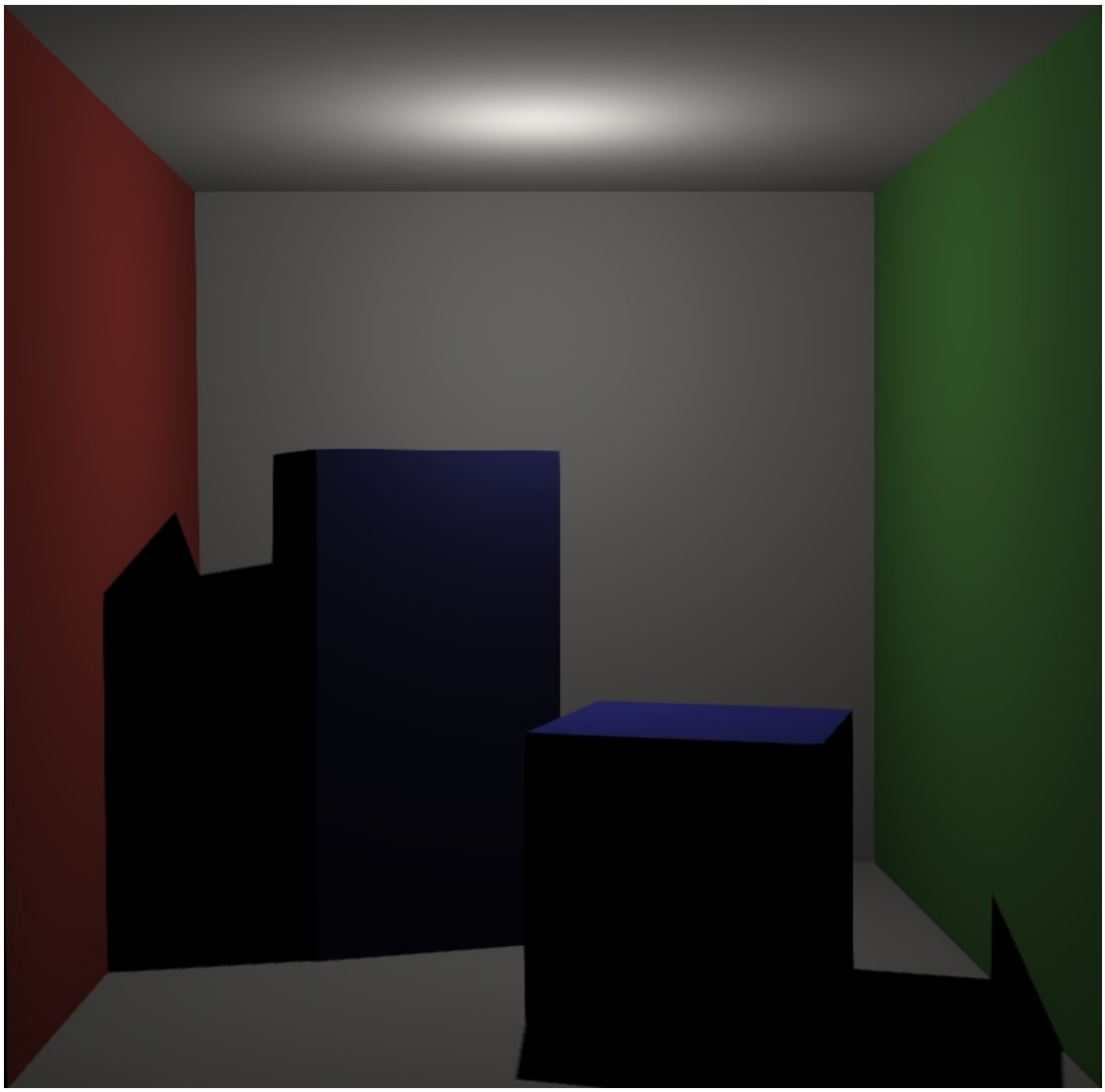


Figure 5.1 – Boite de Cornell, illumination directe.

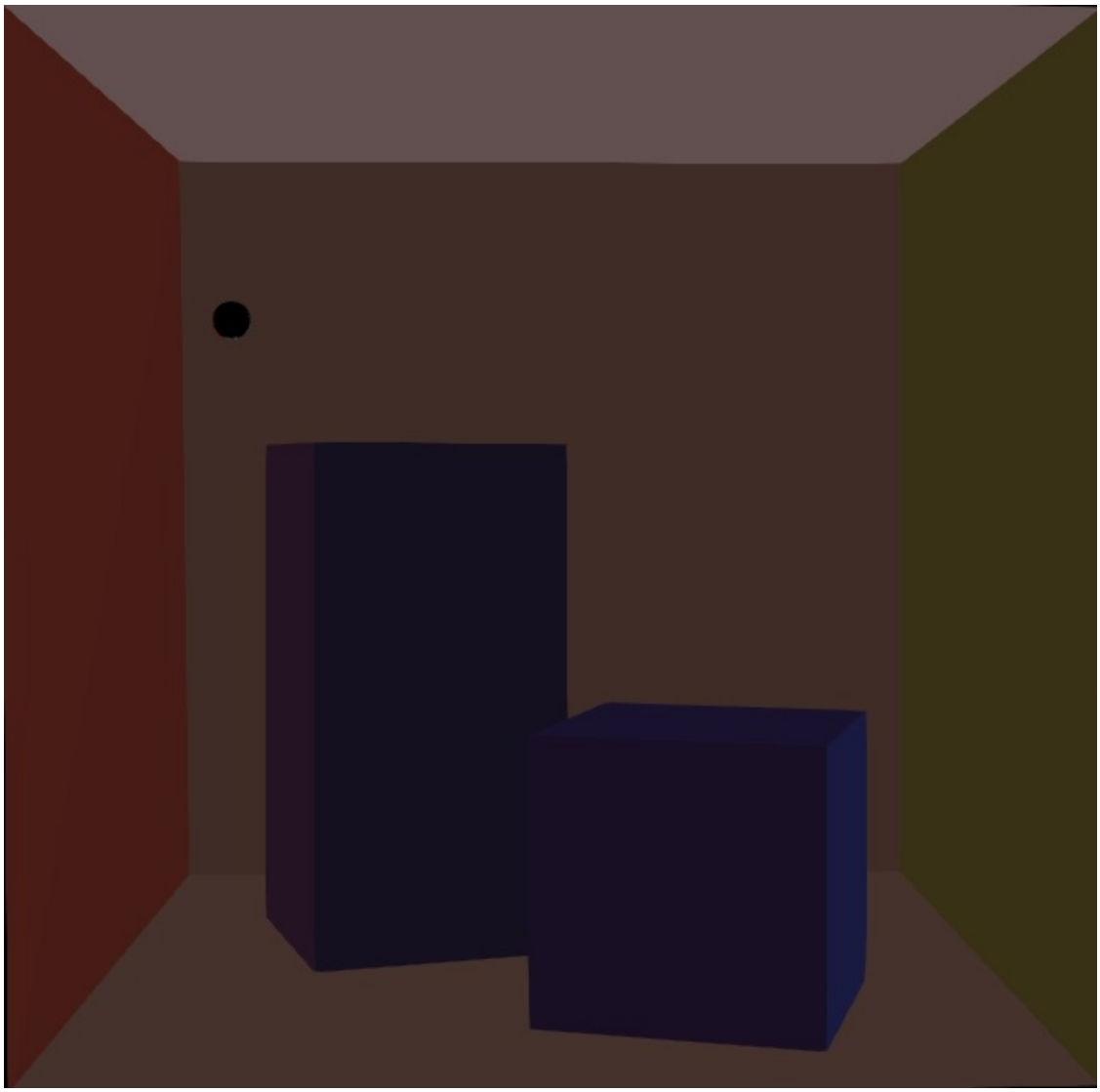


Figure 5.2 – *Shading* produit par un probe placé au hasard, Boite de Cornell.

$$\vec{dp} = \begin{bmatrix} 0.0757511 \\ 0.341979 \\ -0.681559 \end{bmatrix}.$$

On s'attendrait à ce que le vecteur pointe franchement vers le centre de la scène ; ce n'est pas tout à fait ce qui se produit, alors que le probe tente de descendre l'axe vertical et de s'éloigner sur l'axe z. Cependant, à la nouvelle position, l'erreur passe à 1.218, une diminution d'environ 0.012%. Ce n'est pas une bien grande diminution, mais elle suggère que le processus peut fonctionner. En produisant un grand nombre d'itérations successives et en laissant le probe suivre la trajectoire donnée, on obtient le résultat qu'on peut voir à la figure 5.3.

C'est une réussite ! Bien que le probe a pris un détour, il a fini par réussir à aller se placer à la position de l'échantillon. Le tableau 5.4 montre l'évolution de la fonction d'erreur à chaque itération. Cette diminution monotone de la fonction d'erreur est une réussite en soit ; elle suggère que nous sommes prêts à tenter notre chance à une réelle application de la méthode, soit avec un nombre d'échantillons significatif. Des tests subséquents avec des nombres croissants de probes et d'échantillons nous ont convaincus du bien-fondé de notre raisonnement.

5.2 Boite de Cornell

Avec le résultat précédent, nous avons confirmé que la seconde phase de la méthode développée au chapitre précédent fonctionne : il est possible d'utiliser les gradients des coefficients SH pour faire correspondre une structure de probes à un petit nombre d'échantillons. L'étape suivante est d'appliquer le processus en entier.

5.2.1 Référence trilinéaire

Le but de notre étude étant de faire mieux automatiquement qu'une structure de probes comparative placée à la main, nous commençons par en construire une telle. Nous avons choisi de construire nos références comme des structures de probes placées

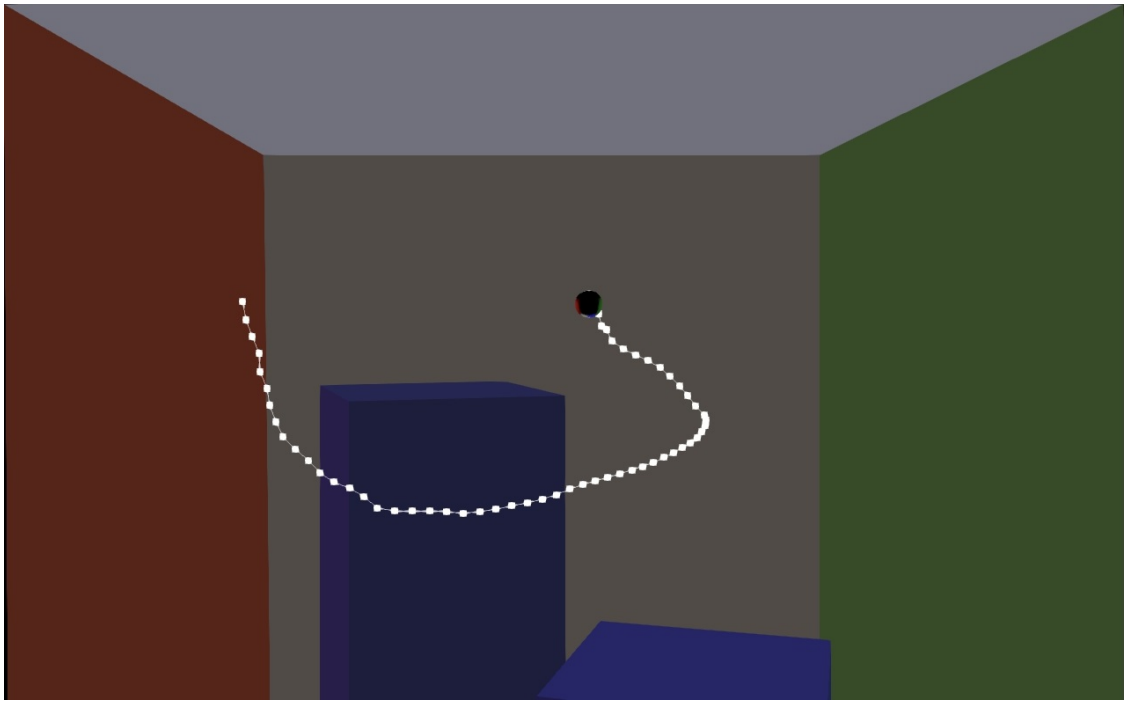


Figure 5.3 – Trajectoire du probe avec un échantillon, Boite de Cornell.

sur une grille interpolées trilinéairement.

Seulement la contribution des probes est représentée ici. On constate que les interréflections entre les murs colorés et blancs sont capturées, tel que souhaité. Nous avons calculé l'erreur produite par cette structure pour différents ensembles d'échantillons (tableau 5.I).

On remarque que la structure offre une meilleure performance avec un grand nombre d'échantillons. La contribution de chaque échantillon à l'erreur moyenne devient de plus

Nombre d'échantillons	Erreur	Erreur moyenne par échantillon
10	5.85	0.585
25	9.67	0.387
50	16.75	0.335
100	35.48	0.354
500	136.61	0.273
1000	310.92	0.311

Tableau 5.I – Erreur produite par la structure de référence, Boite de Cornell.

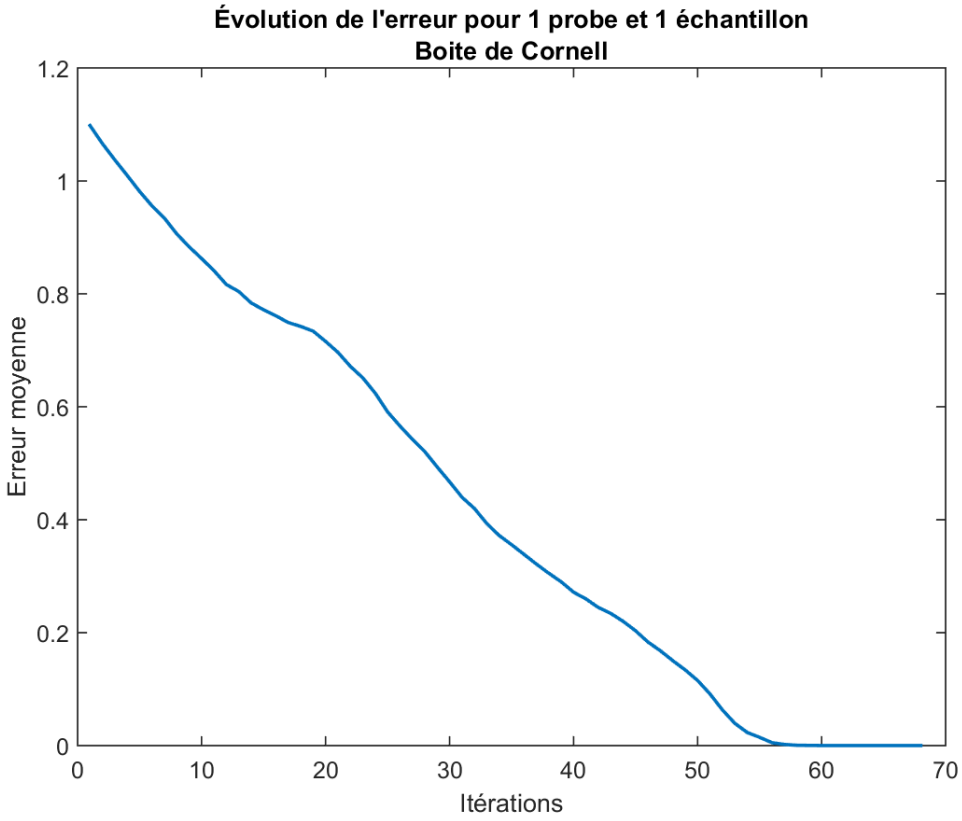


Figure 5.4 – Évolution de la fonction d'erreur pour un probe et un échantillon, Boite de Cornell.

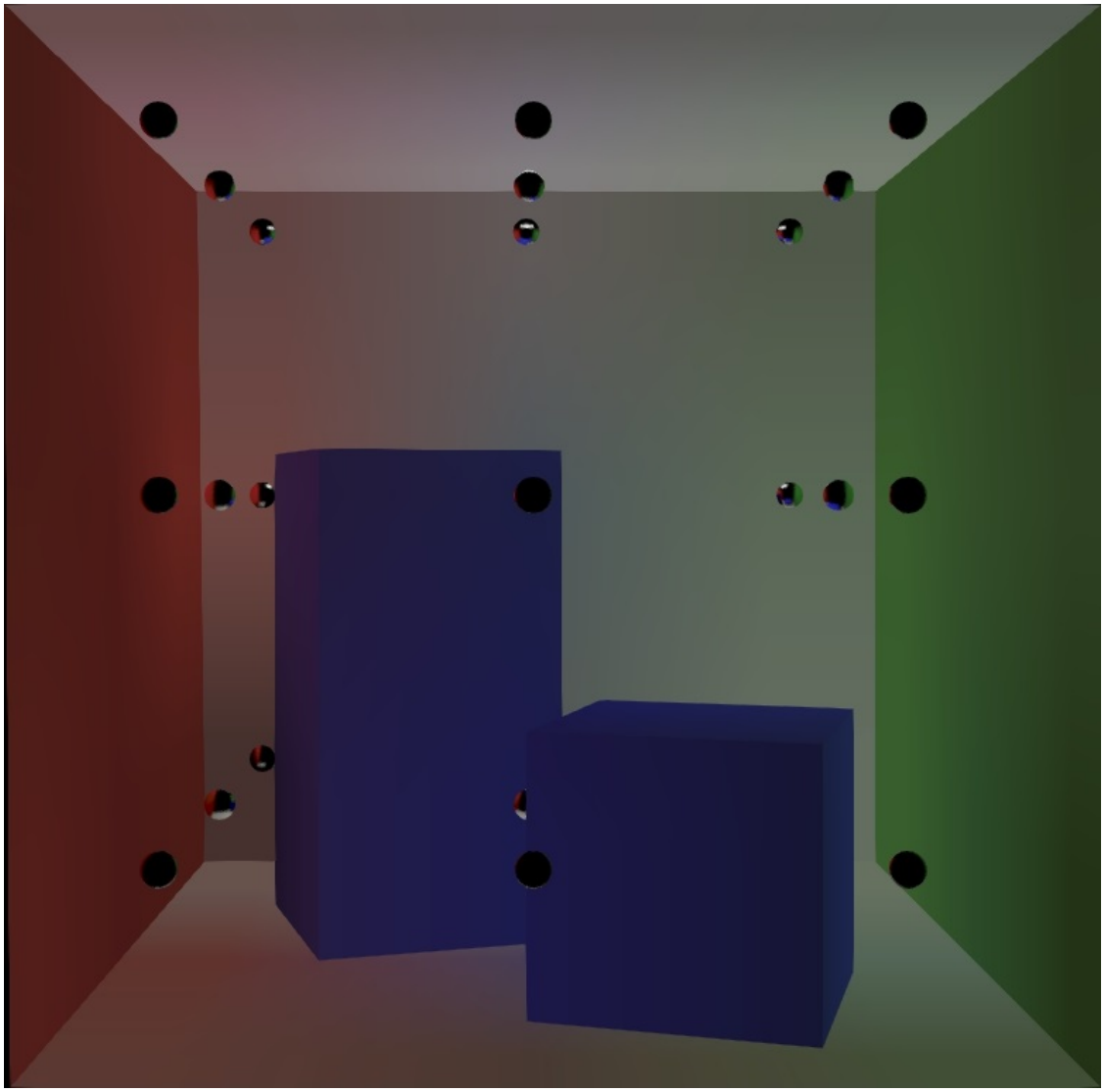


Figure 5.5 – Structure de référence, Boite de Cornell.

en plus petite à mesure que leur nombre augmente ; ainsi l'effet d'avoir un échantillon mal reconstruit devient de plus en plus petit, et l'erreur moyenne se stabilise.

5.2.2 Construction

Nous allons procéder à l'application de notre méthode pour la boite de Cornell. D'abord, voyons les diverses étapes du processus, en prenant l'essai utilisant 100 échantillons comme exemple.

L'application de notre méthode requiert de placer un probe initial, puis de laisser l'optimiseur le déplacer jusqu'à ce qu'il ne puisse plus faire baisser la fonction d'erreur. Plusieurs centaines de positions distribuées aléatoirement dans la scène sont testées, et le probe offrant la meilleure performance est gardé.

Une fois la position optimale obtenue, on applique la phase 2, qu'on peut visualiser sur les figures 5.6 et 5.7.

On remarque qu'un probe unique tend à aller se placer autour du milieu de la scène ; ce comportement est dû au fait qu'il tente de répartir la distribution des composantes du terme d'erreur sur tous les échantillons dans la scène. Ceux-ci étant répartis uniformément, le probe initial se dirige vers la position de compromis maximal pour tous les échantillons.

L'erreur par échantillon passe de 0.98 à 0.85 au cours de ce processus, et après quelques itérations, l'optimiseur ne peut en produire une meilleure. La figure 5.8 montre l'ajout du second probe, suite à la seconde exécution de la phase 1.

On répète le processus jusqu'à obtenir le nombre de probes désiré selon le budget, ou que l'ajout de nouveaux probes ne vienne pas améliorer le résultat.

Après un peu plus de 200 itérations, la variation de l'erreur est stabilisée, et l'ajout de nouveaux probes ne vient pas apporter de piste pour que l'optimiseur puisse la faire baisser de façon significative. La figure 5.10 montre la progression de la fonction d'erreur au cours du processus.

On identifie bien, dans cette courbe, les étapes de notre processus. Les barres verticales correspondent à l'ajout d'un probe, qui vient généralement faire baisser la valeur courante de l'erreur. Entre les ajouts de probes se suivent des descentes plus lisses qui

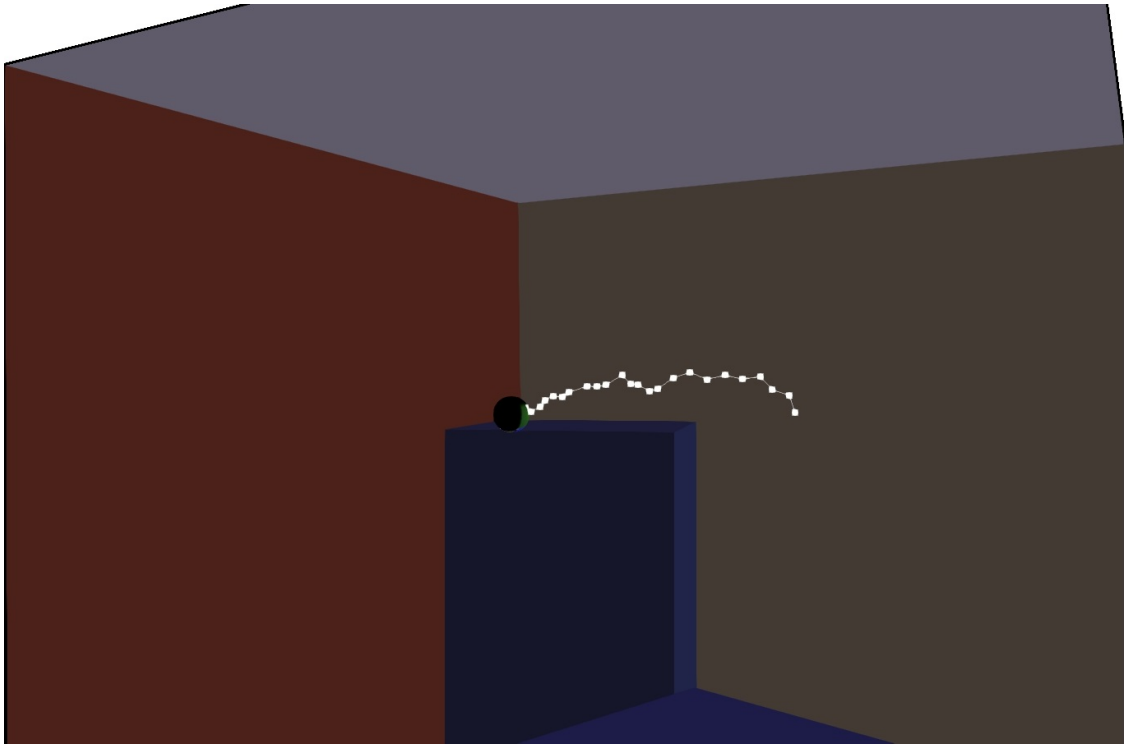


Figure 5.6 – Trajectoire pour la première exécution de la phase 2, Boite de Cornell.

correspondent à la deuxième phase. La courbe qui stagne trop ou qui se met à remonter déclenche l'ajout d'un autre probe.

Bien que l'application de la technique ait fonctionné, nos attentes ne sont pas entièrement comblées : nous n'avons pas réussi à faire mieux que la structure de probes trilinéaire, avec une erreur finale presque deux fois plus grande. Ce constat est contre-balancé par l'observation que notre structure construite comporte moins de la moitié du nombre de probes de la structure de référence et qu'elle a été construite entièrement sans intervention humaine.

L'exemple précédent était pour 100 échantillons, mais nous avons testé différentes tailles d'ensembles d'échantillons pour jauger le comportement de notre méthode. La figure 5.11 montre l'évolution de la fonction d'erreur pour les différents nombres d'échantillons utilisés. Le tableau 5.II montre les meilleures valeurs d'erreur obtenues au cours des optimisations et le nombre de probes correspondant.

On constate que les résultats obtenus par notre méthode vont à l'inverse de ceux

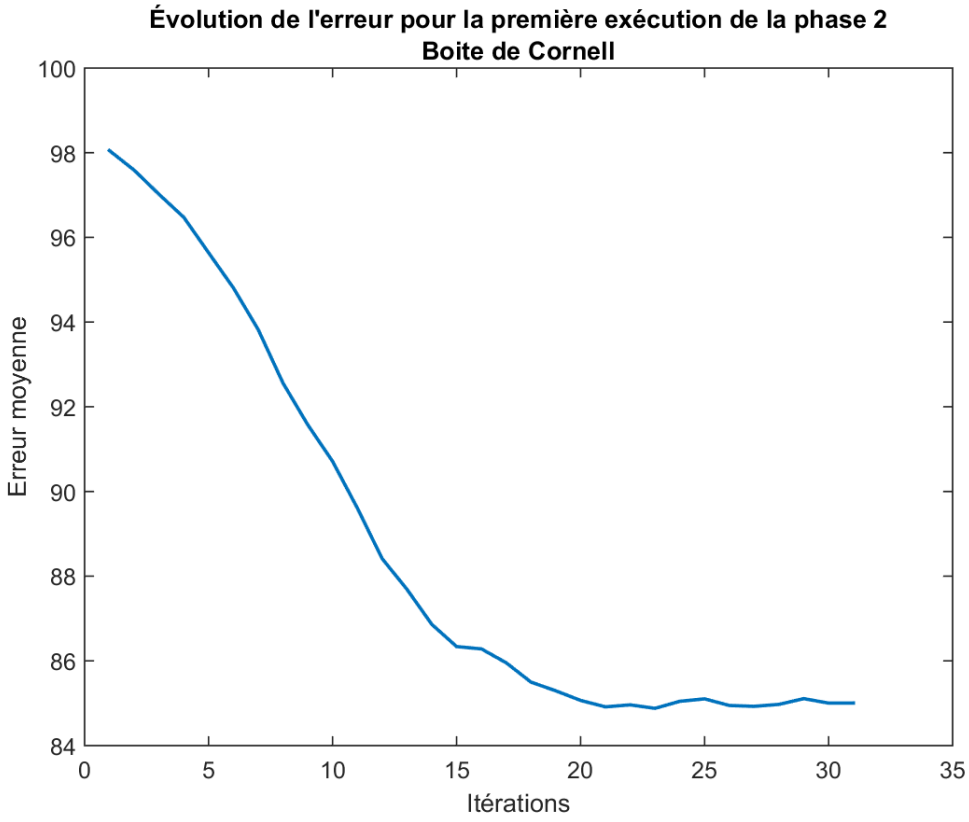


Figure 5.7 – Évolution de l'erreur pour la première exécution de la phase 2, Boite de Cornell.

Nombre d'échantillons	Erreur minimale	Erreur minimale moyenne par échantillon
10	0.389	0.039
25	7.12	0.285
50	17.96	0.359
100	44.90	0.449
500	247.36	0.498
1000	486.81	0.487

Tableau 5.II – Résultats du processus d'optimisation selon le nombre d'échantillons, Boite de Cornell.

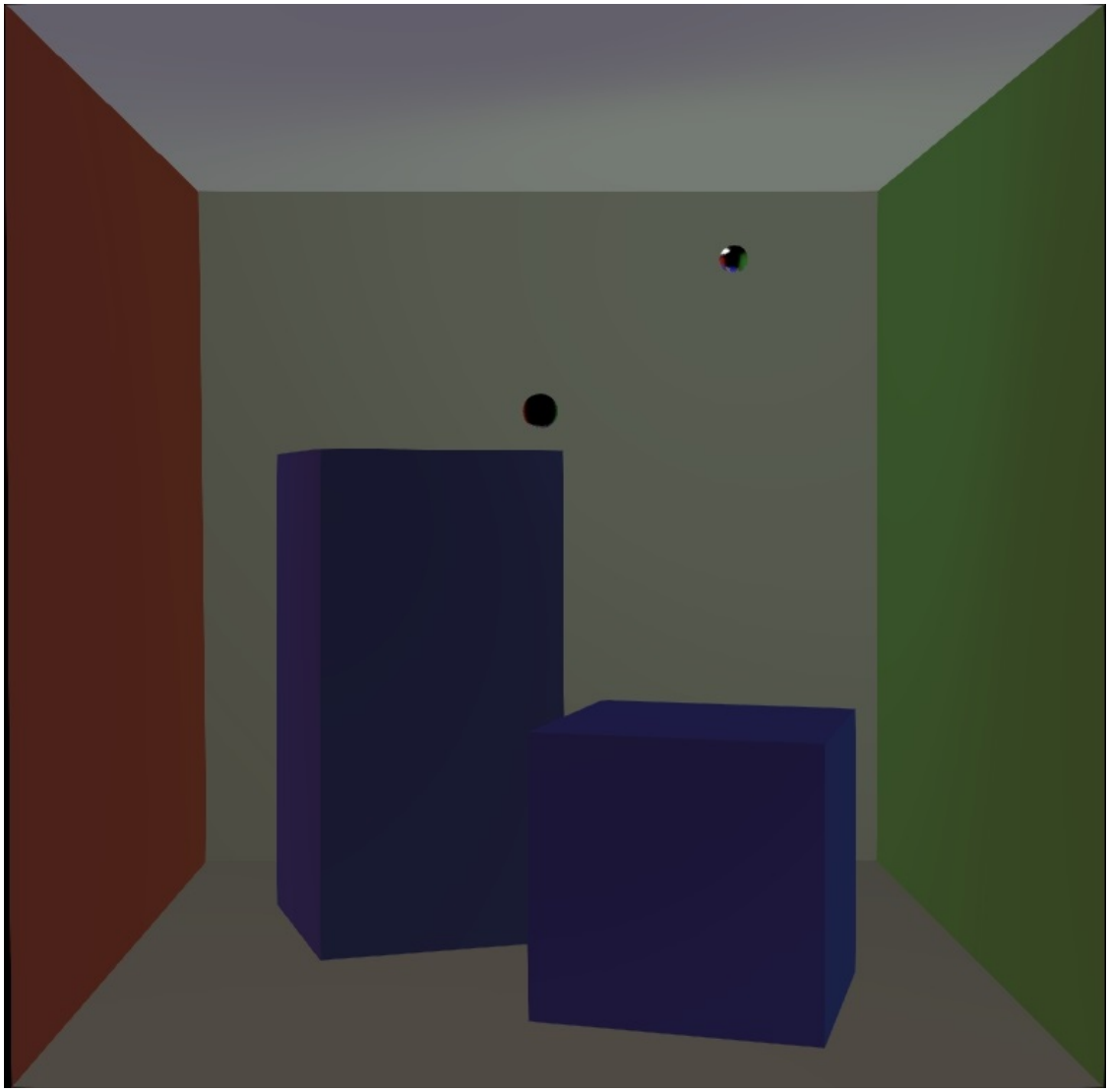


Figure 5.8 – Ajout d'un probe (répétition de la phase 1), Boite de Cornell.

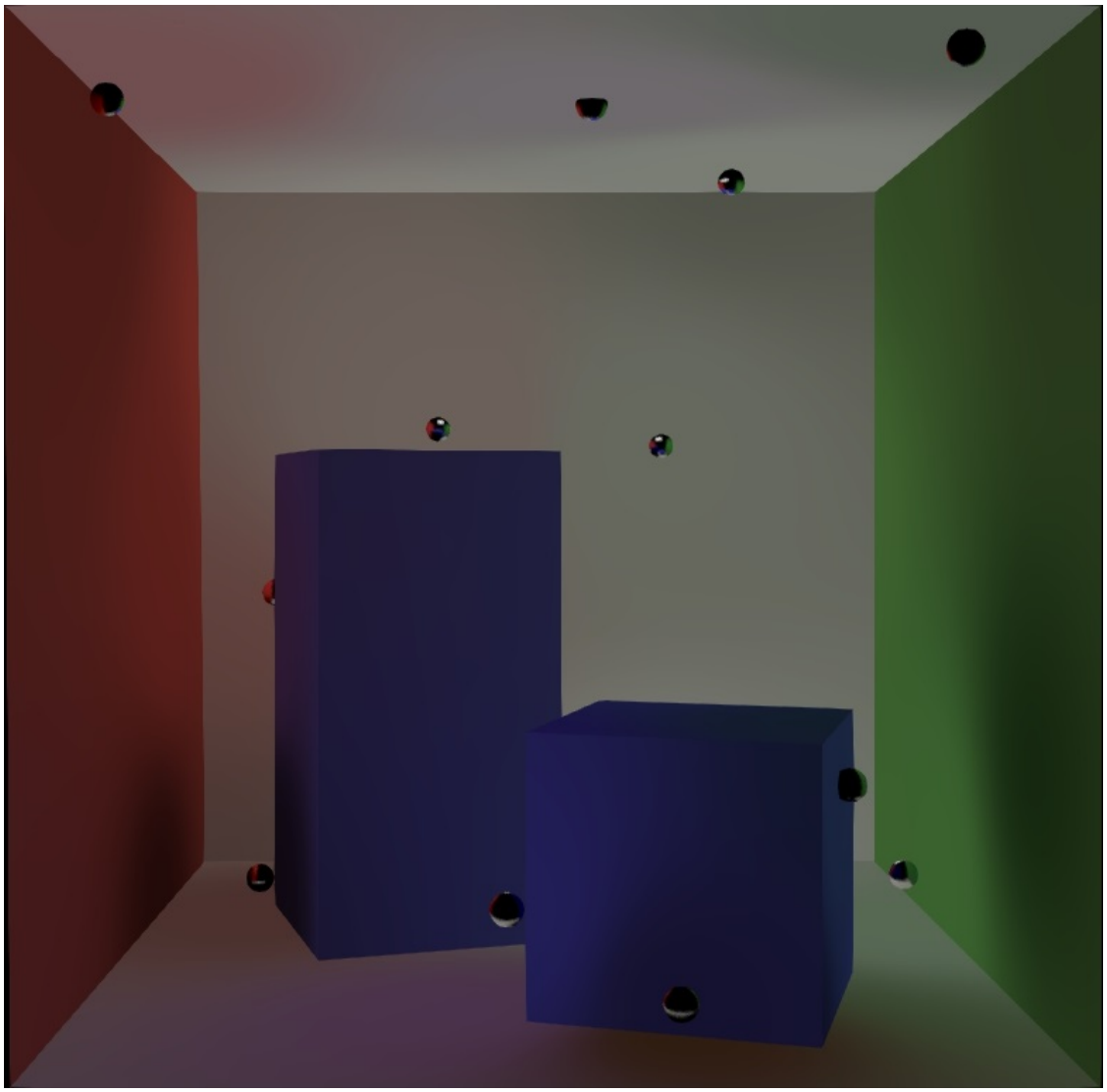


Figure 5.9 – Résultat du processus d'optimisation, 100 échantillons, Boite de Cornell.

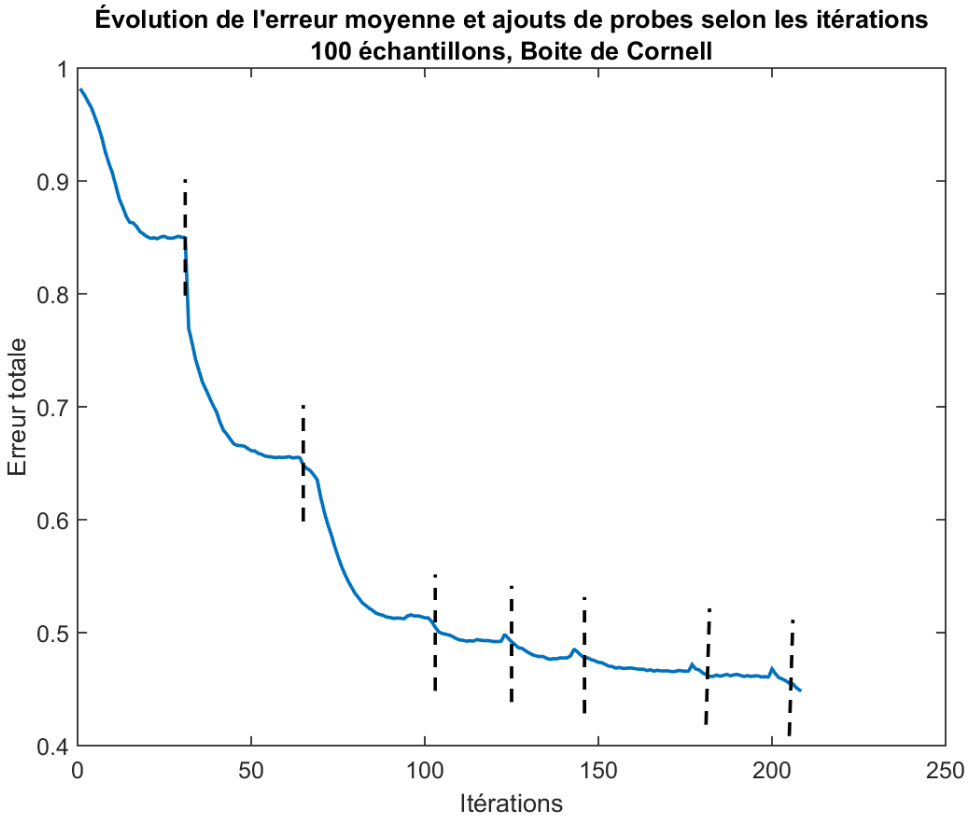


Figure 5.10 – Évolution de l'erreur, 100 échantillons, Boite de Cornell.

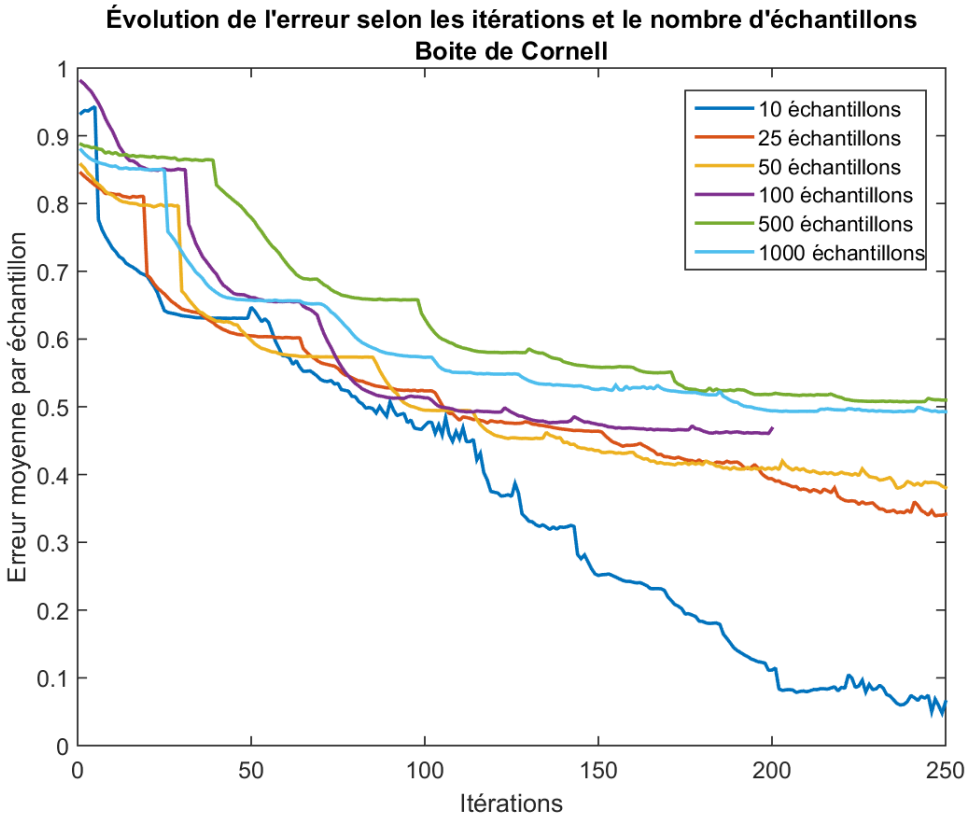


Figure 5.11 – Résultats du processus d'optimisation selon le nombre d'échantillons, Boite de Cornell.

fournis par la structure trilinéaire de référence. Pour 10 échantillons, rappelons que l'erreur par échantillon était de 0.585 pour la structure de référence. Notre méthode arrive à produire un score similaire avec seulement 3 probes ! Il s'agit donc d'une réduction du nombre de probes nécessaire d'un facteur 9. On remarque également que la plus petite valeur d'erreur que l'optimiseur réussit à atteindre est 0.04 par échantillon, ce qui est très petit ; à ce point la structure comporte 14 probes, soit plus que le nombre d'échantillons considéré.

On constate qu'à partir de 100 échantillons, l'erreur de reconstruction dépasse l'erreur de la structure de référence. Il semble donc qu'une structure trilinéaire soit plus adaptée à une haute densité d'échantillons, au moins pour cette scène, où la lumière n'est pas distante, et où la visibilité est obstruée par les boites. Malgré ceci, il est satisfaisant de voir que notre algorithme arrive à faire descendre l'erreur de façon régulière.

On remarque que la première phase ne mène pas nécessairement à une erreur plus basse, et qu'un compromis doit parfois être fait : accepter que l'erreur augmente légèrement à l'ajout d'un probe, en espérant que l'algorithme de minimisation réussira à la faire redescendre par après.

Bien que cette scène soit très simple, il s'agit d'un bon point de départ ; voyons comment la technique se comporte pour une scène plus complexe.

5.3 Atrium de Sponza

Nous avons choisi l'atrium de Sponza [26] comme scène plus avancée, présentée à la figure 5.12.

Cette scène est grande et complexe ; elle comporte plusieurs "sections" non-visibles les unes des autres. Nous avons choisi un éclairage distant provenant du soleil et baignant de lumière la moitié de la cour intérieure ; ainsi, la plupart de la scène est éclairée de façon indirecte.

Voyons d'abord l'objectif à battre.

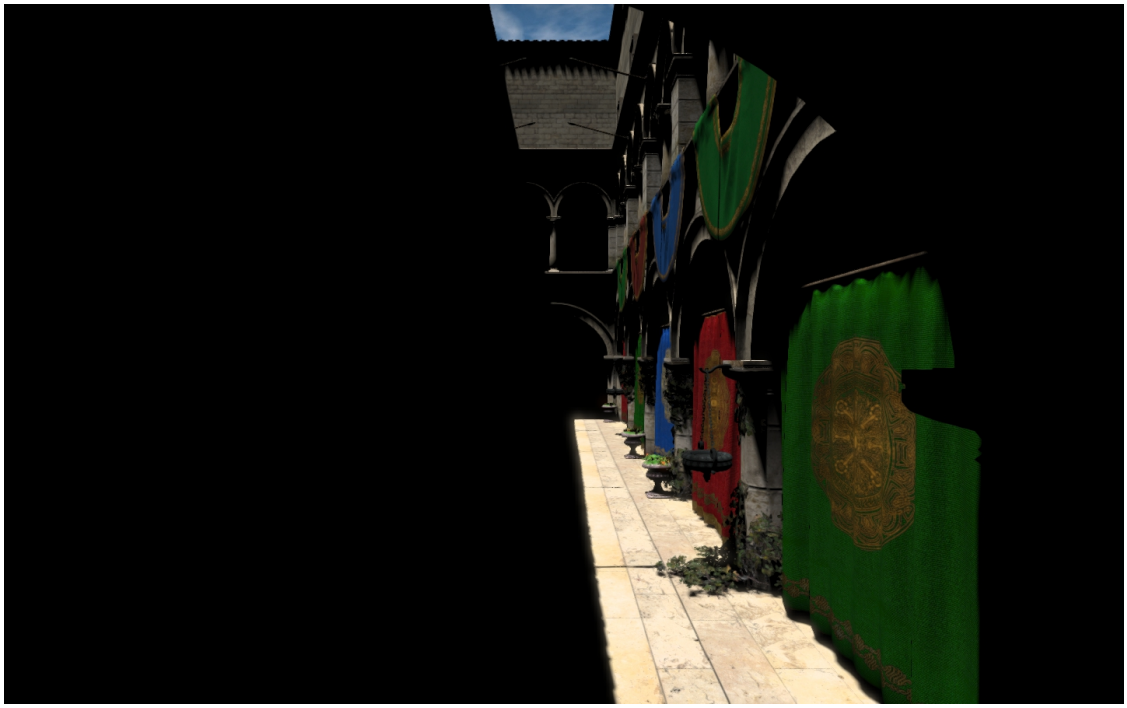


Figure 5.12 – Atrium de Sponza, éclairage direct seulement.

5.3.1 Référence trilinéaire

Cette scène étant beaucoup plus grande que la boite de Cornell, le désavantage apporté par une grille régulière commence à se faire voir. Pour bien couvrir toute la scène, nous avons choisi un espacement d'un peu plus de 3 unités entre les probes ; ce choix exige quand même, pour une couverture maximale, d'utiliser une structure comportant $10 \times 6 \times 4$ probes, pour un total de 240. Les figures 5.13, 5.14 et 5.15 montrent respectivement la somme de l'éclairage direct et indirect, la composante indirecte seulement puis la composante indirecte de l'éclairage sans texture afin de bien mettre en évidence la couleur des interréflections. Les probes situés devant les draperies apportent bien cette contribution.

Le tableau 5.III montre quant à lui les erreurs de reconstruction de la référence pour divers nombres d'échantillons testés.

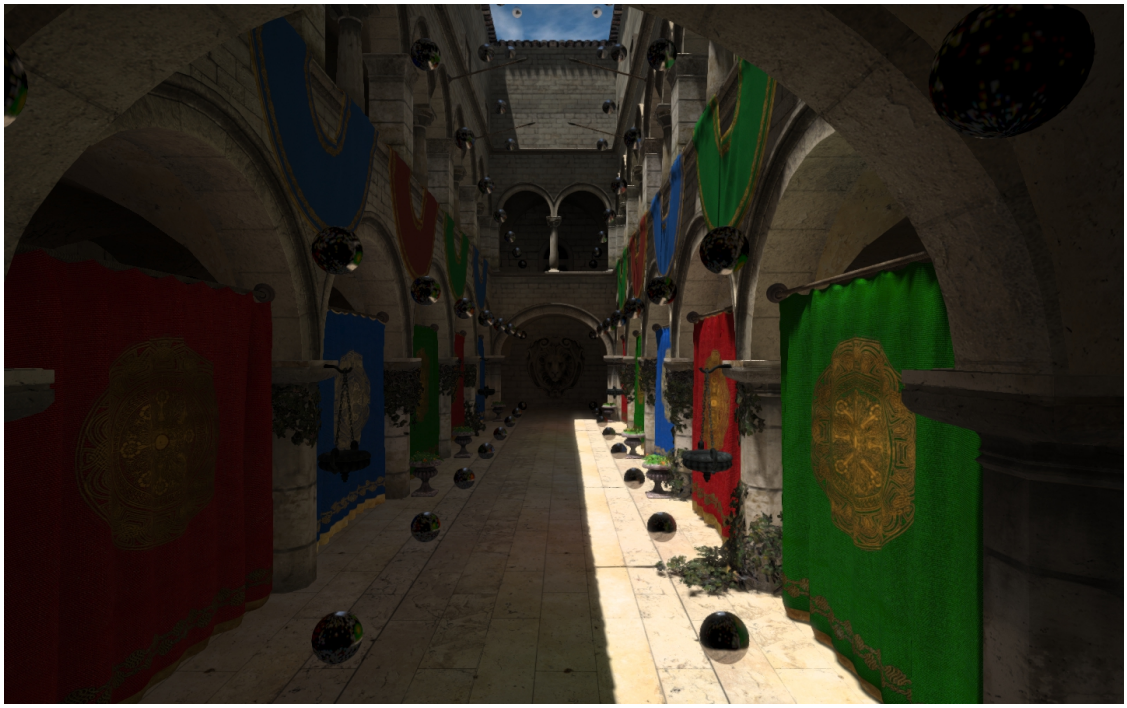


Figure 5.13 – Éclairage direct + indirect des probes de référence, Atrium de Sponza.

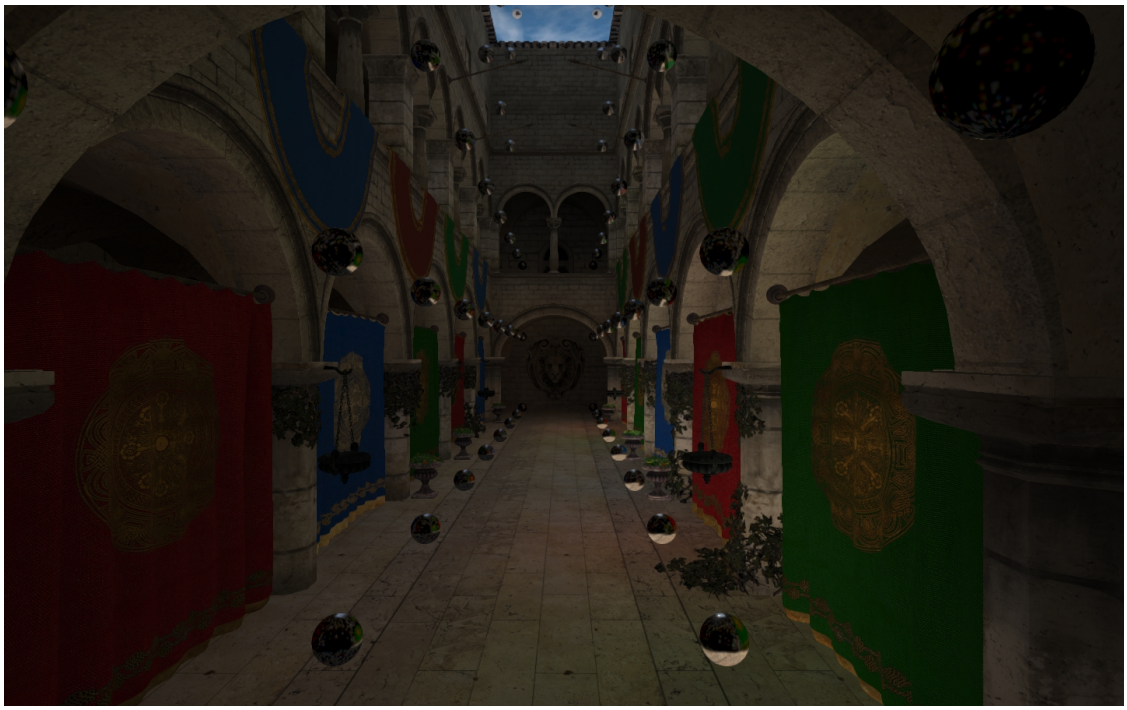


Figure 5.14 – Éclairage indirect des probes de référence, Atrium de Sponza.

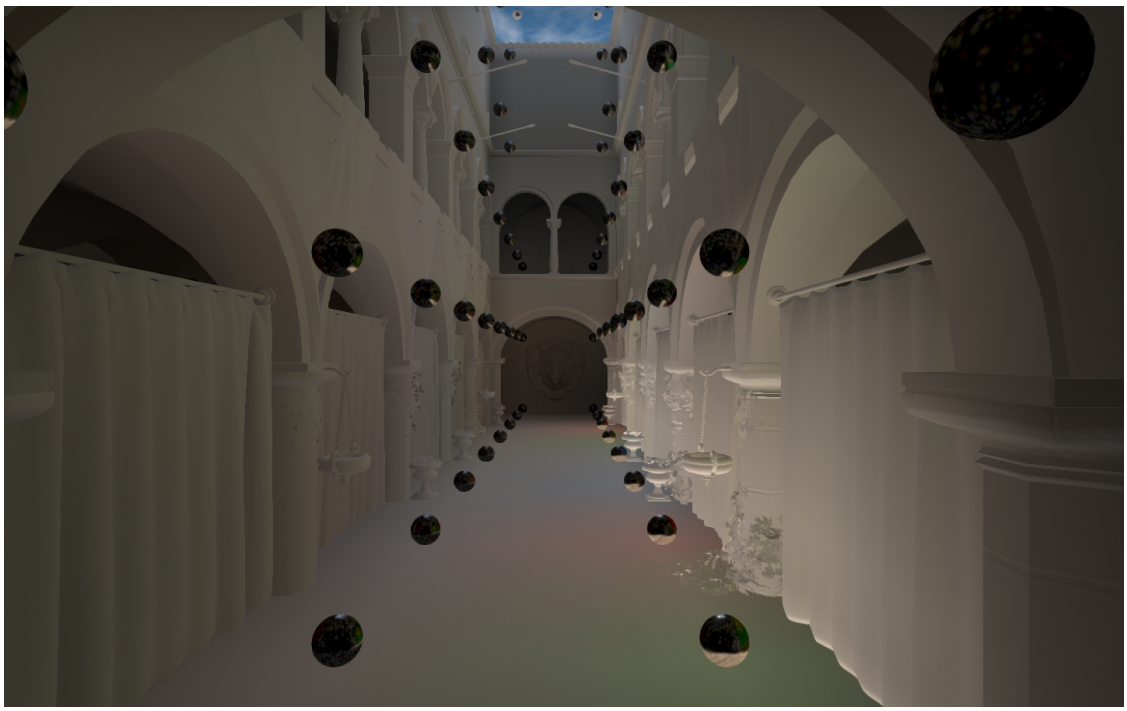


Figure 5.15 – Éclairage indirect sans texture, Atrium de Sponza.

Nombre d'échantillons	Erreur	Erreur moyenne par échantillon
10	2.696	0.269
25	5.355	0.214
50	9.207	0.184
100	37.268	0.372
500	119.64	0.239
1000	240.24	0.240
2000	465.64	0.231

Tableau 5.III – Erreur produite par la structure de référence selon le nombre d'échantillons, éclairage distant, Atrium de Sponza.

5.3.2 Construction

La reconstruction dans Sponza est plus laborieuse, et les résultats obtenus moins satisfaisants. La figure 5.16 et le tableau 5.IV montrent les résultats.

Ces optimisations ont été arrêtées puisque l'optimiseur stagnait et n'arrivait plus à faire baisser l'erreur de façon satisfaisante. On constate que dans la majorité des cas, on ne réussit pas à battre l'erreur de la référence ; les essais avec 10 et 25 échantillons ont abouti à des meilleures valeurs d'erreur, mais au prix de grands nombres de probes par rapport au nombre d'échantillon. À partir de 50 échantillons, on ne peut même plus battre la structure de référence.

5.4 Salon

La dernière scène que nous avons testée se veut un compromis entre les deux scènes précédentes ; nous avons choisi une configuration plus simple que l'atrium de Sponza, mais plus grande et plus aérée que la boite de Cornell.

Cette scène [10] représente simplement un salon ; les murs bleus et le plancher en bois contribuent aux interréflexions indirectes. Nous en avons également profité pour tester deux environnements d'éclairage : un où seule la lumière du soleil entre par une fenêtre, puis un autre où la lumière vient d'une source située au plafond de la pièce.

Nombre d'échantillons	Erreur minimale	Erreur minimale moyenne	Nombre de probes
10	0.136	0.0136	14
25	2.67	0.107	40
50	11.484	0.229	19
100	38.58	0.386	28
500	187.10	0.374	35
1000	365.4	0.365	39
2000	721.334	0.36	35

Tableau 5.IV – Résultats du processus d'optimization selon le nombre d'échantillons, Atrium de Sponza.

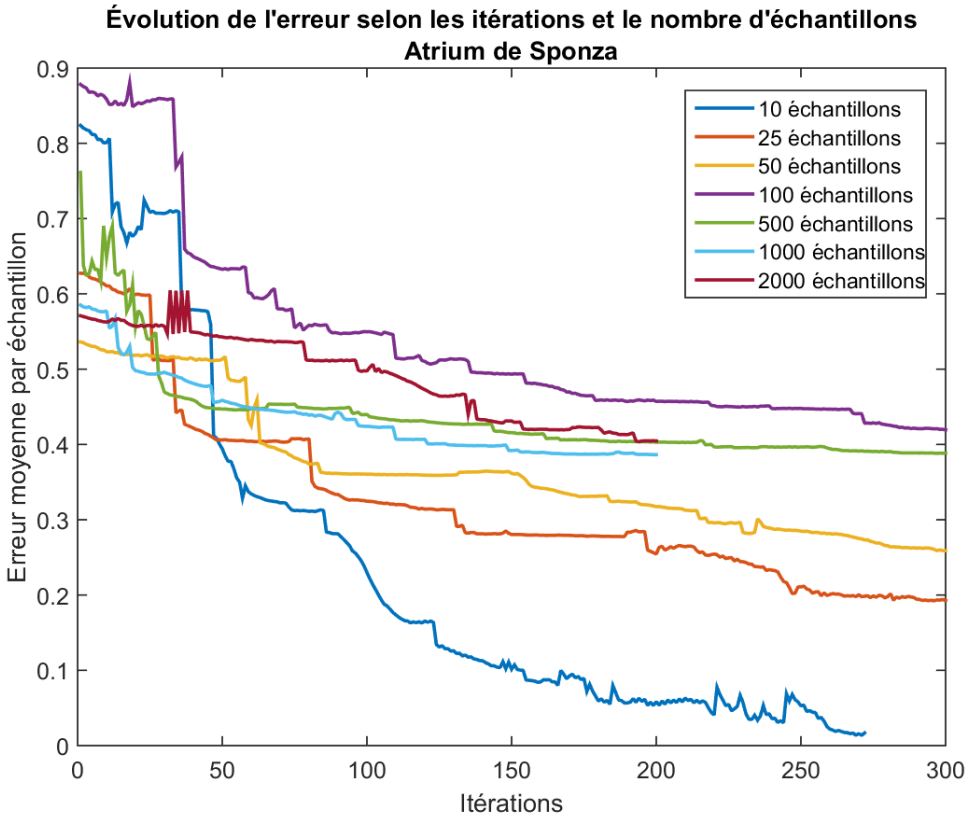


Figure 5.16 – Résultats du processus d'optimisation, Atrium de Sponza.

5.4.1 Éclairage distant

Dans notre premier test, la scène n'est éclairée que par la lumière du ciel et du soleil qui entrent par les fenêtres, ce qui correspond à un éclairage plus distant, et donc plus approprié à notre méthode. Dans ce cas, la majorité de la scène est éclairée par des contributions indirectes. On peut observer la scène sous cet éclairage à la figure 5.17.

5.4.1.1 Références

Une structure de probes trilinéaire de dimensions $3 \times 2 \times 3$, soit 18 probes, représente notre résultat de référence (figures 5.18 et 5.19, tableau 5.V).

Il est intéressant de constater que ces résultats représentent la meilleure erreur moyenne par échantillon vue jusqu'à maintenant. Il semble que cette scène soit mieux adaptée à notre modèle ; sa simplicité géométrique par rapport à la boite de Cornell et Sponza explique probablement ceci.

5.4.1.2 Construction

Notre algorithme offre de bonnes performances au niveau de la reconstruction. Nous avons coupé l'exécution plus vite cette fois, préférant s'arrêter après seulement 15 probes, peu importe le nombre d'échantillons choisi ; l'erreur cesse de descendre assez pour justifier l'ajout de nouveaux probes. On peut voir les résultats à la figure 5.20 et au tableau 5.VI.

Il est satisfaisant de constater que dans tous les cas, l'erreur finale est plus basse que

Nombre d'échantillons	Erreur	Erreur moyenne par échantillon
10	1.898	0.189
25	3.88	0.155
50	5.31	0.106
100	11.164	0.112
500	54.57	0.109
1000	114.94	0.115

Tableau 5.V – Erreur produite par la structure de référence selon le nombre d'échantillons, Salon, éclairage distant.

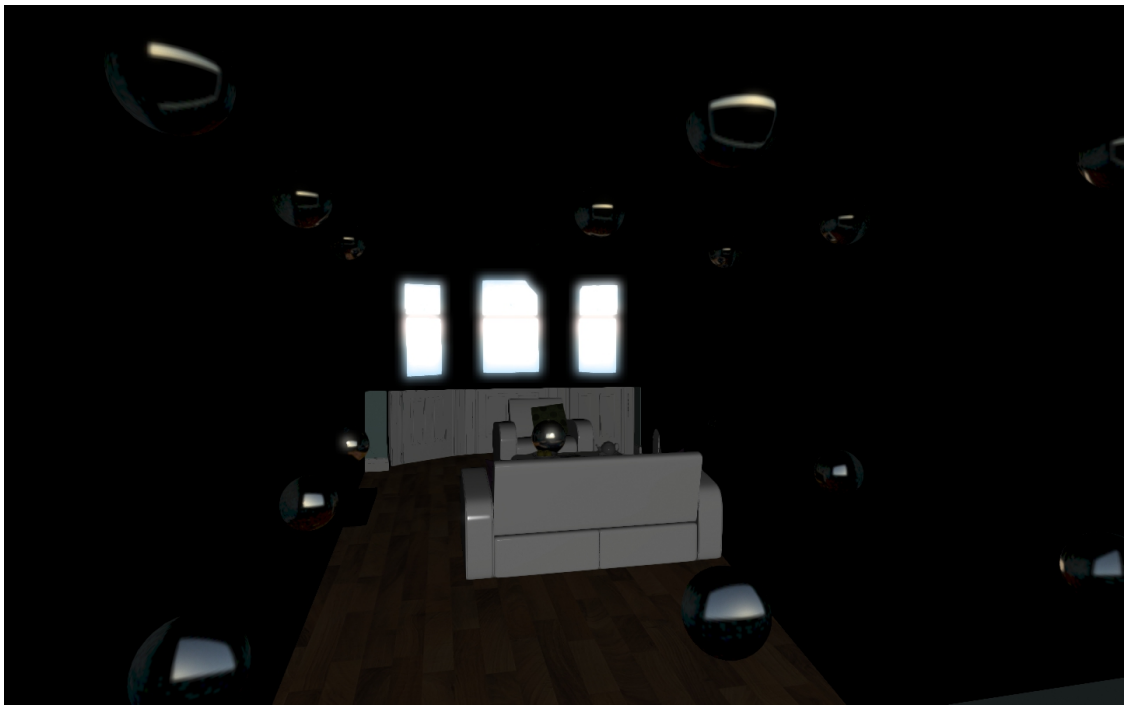


Figure 5.17 – Salon, structure de référence, éclairage distant, composante directe seulement.

Nombre d'échantillons	Erreur minimale	Erreur minimale moyenne	Nombre de probes
10	0.163	0.0163	11
25	1.11	0.044	9
50	2.46	0.049	12
100	6.342	0.0634	14
500	38.77	0.077	14
1000	79.8	0.079	14

Tableau 5.VI – Résultats du processus d'optimisation selon le nombre d'échantillons, Salon, éclairage distant.

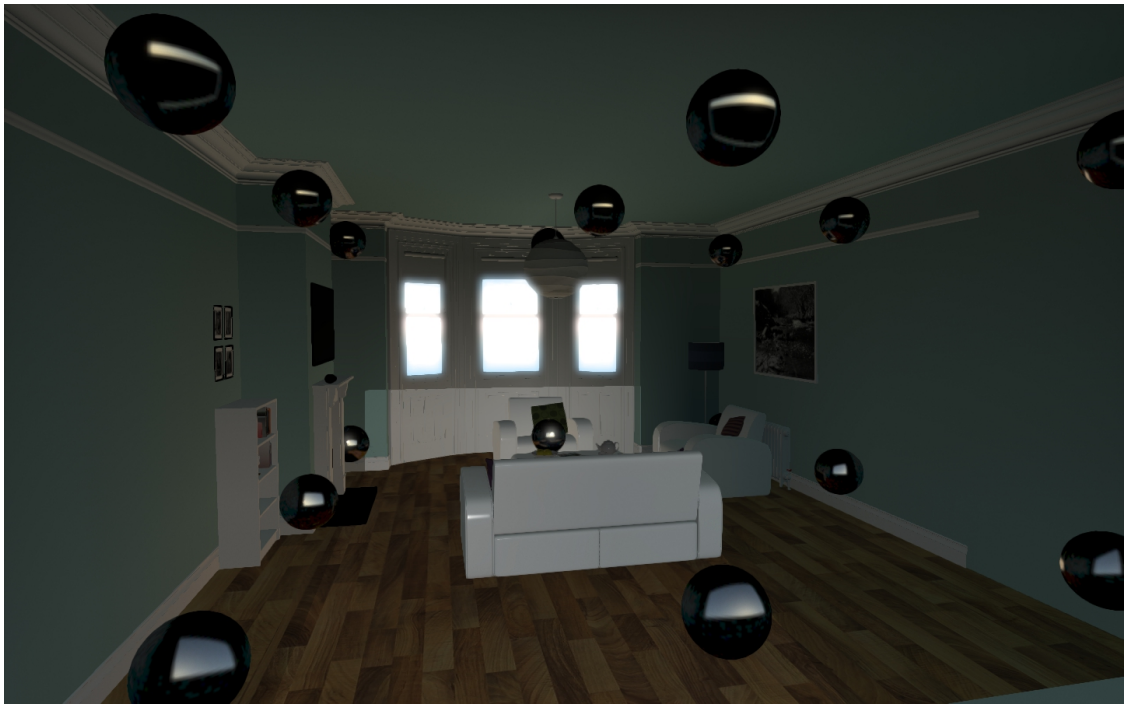


Figure 5.18 – Salon, structure de référence, éclairage distant, composante directe + indirecte venant des probes.

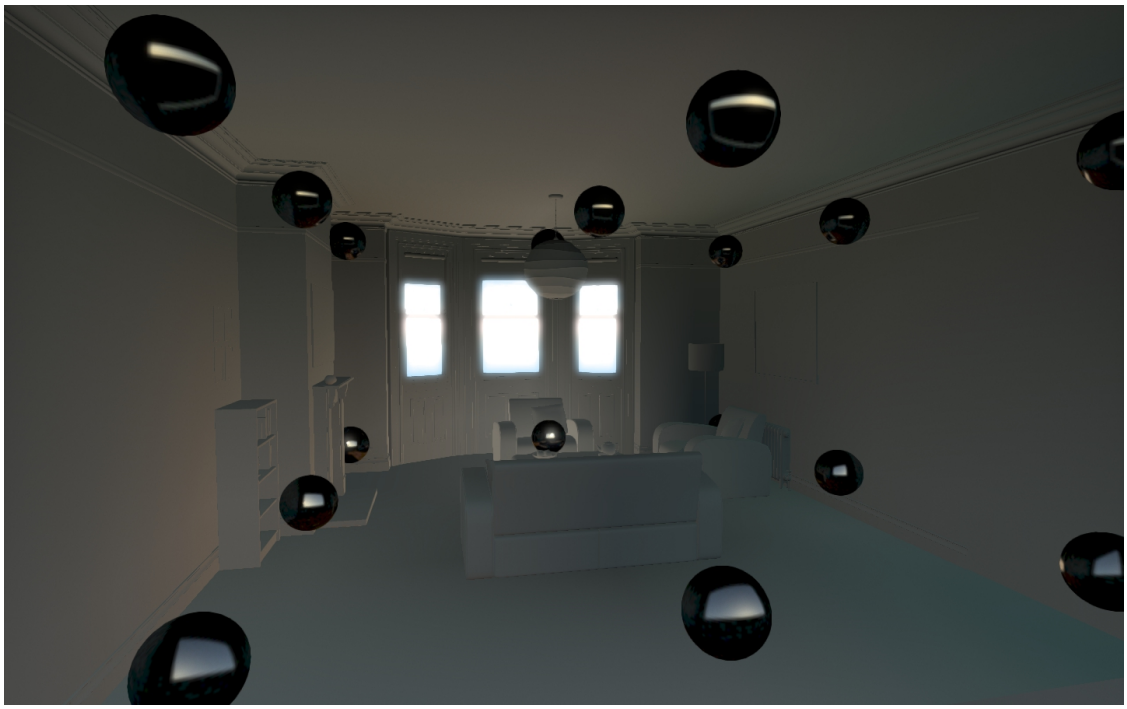


Figure 5.19 – Salon, structure de référence, éclairage distant, sans texture.

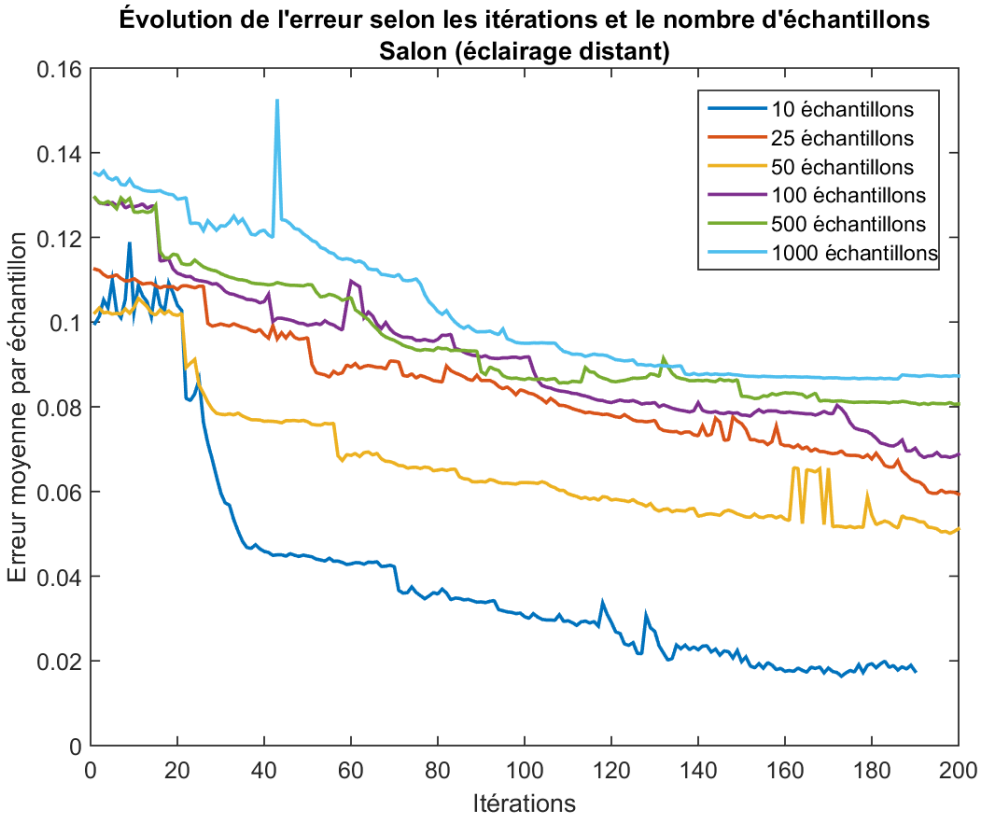


Figure 5.20 – Résultats de l'optimisation, Salon, éclairage distant.

celle produite par la structure de référence. Le tableau 5.VII montre combien de probes sont nécessaires pour *égaliser* cette erreur.

Pour un petit nombre d'échantillons (10-50), un seul probe judicieusement placé peut produire un meilleur résultat que l'interpolation de 18 probes placés sur une grille trilinéaire. À mesure que le nombre d'échantillons augmente, on constate qu'il faut un peu plus de travail, mais il suffit tout de même de deux ou trois probes et de quelques dizaines d'itérations pour obtenir un résultat semblable.

5.4.2 Éclairage proche

Voyons maintenant comment se comporte la scène avec notre version alternative des conditions lumineuses. Nous avons choisi un éclairage proche, où la lumière est fournie par une source au plafond, afin de voir à quel point notre résultat dans la boite de Cornell peut être attribuable à la proximité de la source. La figure 5.21 montre la composante directe de cet éclairage.

5.4.2.1 Référence

La même structure de probes qu'à la section précédente a été utilisée pour la référence ; nous n'avons eu qu'à refaire leur rendu et leur projection en SH. Les figures 5.22, 5.23 et 5.24 montrent la structure de référence, tandis que le tableau 5.VIII montre les valeurs d'erreurs produites par cette structure.

On constate immédiatement que les valeurs obtenues sont beaucoup plus grandes que

Nombre d'échantillons	Itération	Nombre de probes
10	0	1
25	0	1
50	0	1
100	22	3
500	39	2
1000	62	3

Tableau 5.VII – Itération et nombre de probes correspondant à partir de laquelle l'erreur des probes optimisés est inférieure à l'erreur de référence, Salon, éclairage distant.

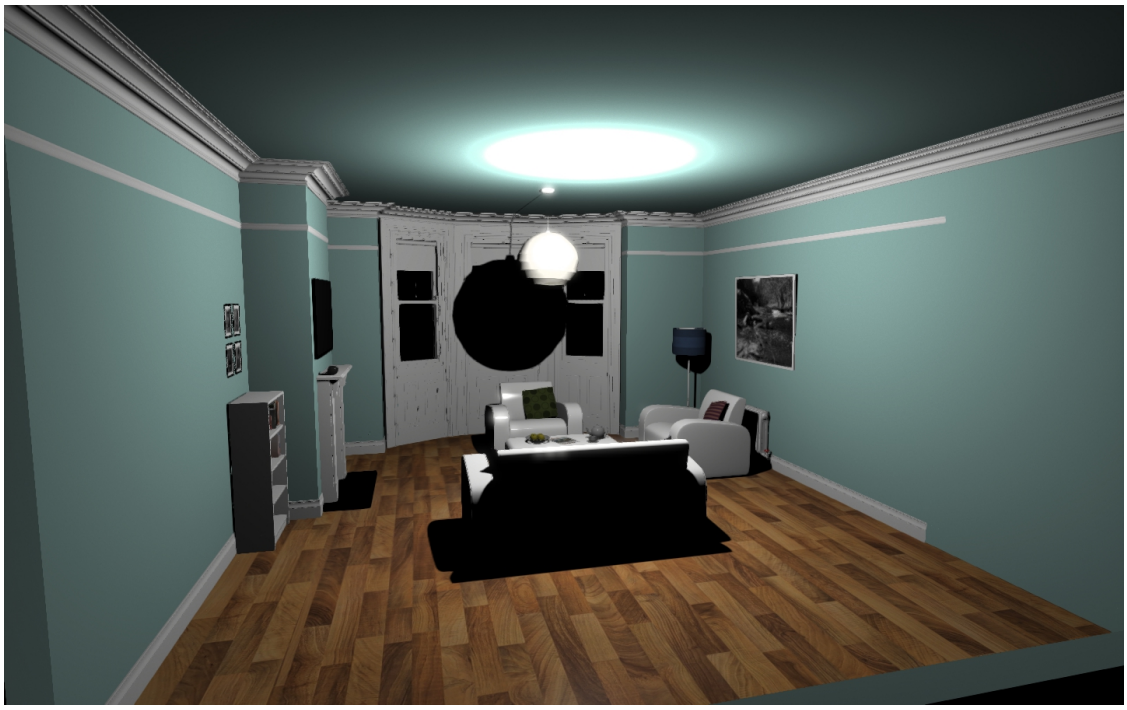


Figure 5.21 – Salon, structure de référence, éclairage proche, composante directe seulement.

Nombre d'échantillons	Erreur	Erreur moyenne par échantillon
10	5.054	0.505
25	11.80	0.472
50	23.87	0.477
100	35.674	0.356
500	180.877	0.362
1000	382.608	0.382

Tableau 5.VIII – Erreur produite par la structure de référence selon le nombre d'échantillons pour la scène du salon, éclairage proche.

Figure 5.22 – Salon, structure de référence, éclairage proche, composante directe + indirecte venant des probes.

celles utilisant l'éclairage distant. On s'attendait à ce résultat, puisque l'expansion en coefficients SH de l'irradiance fonctionne mieux pour des sources de lumière distante ; cette scène étant moins bien adaptée à la technique, plus d'erreur est accumulée.

5.4.2.2 Construction

La figure 5.25 montre l'évolution de l'erreur ; encore une fois, on obtient des résultats satisfaisants.

Une fois de plus, on obtient de bons résultats après 15 probes pour la reconstruction. Encore une fois, le tableau 5.X montre combien de probes sont nécessaires pour avoir une erreur inférieur à la référence.

Encore une fois, il suffit d'un très petit nombre de probes pour obtenir une meilleure erreur de reconstruction que la référence. On atteint ce point moins rapidement qu'avec l'éclairage distant, mais il est tout de même atteint dans un nombre d'itérations raisonnable.

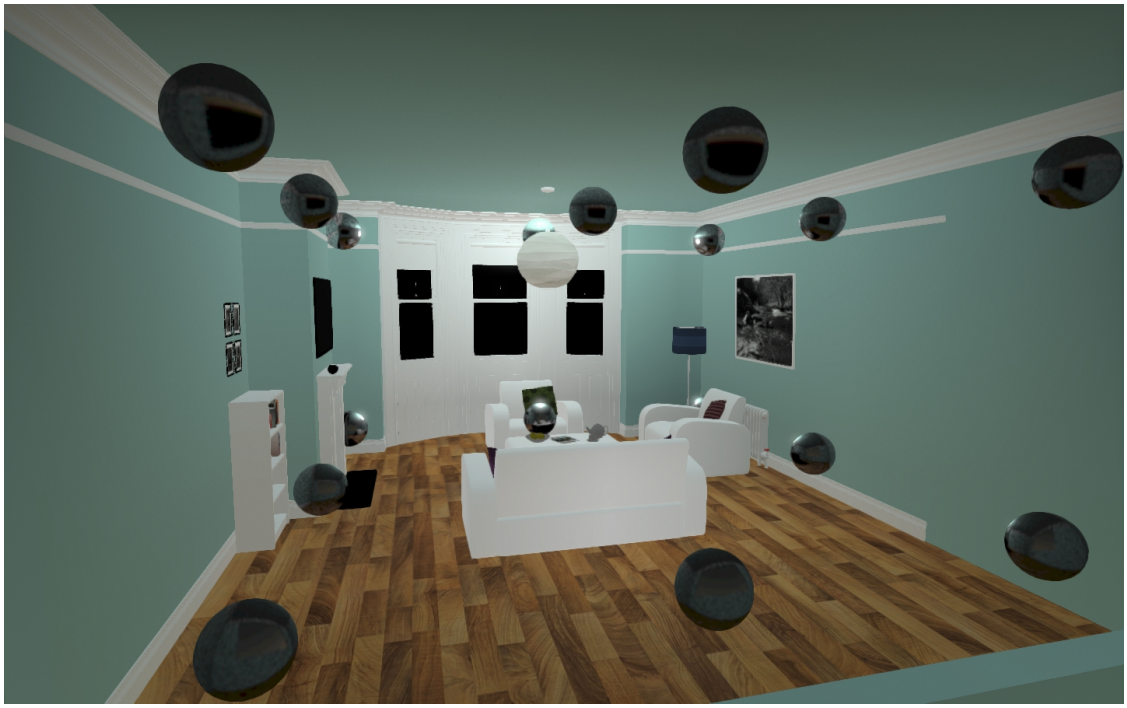


Figure 5.23 – Salon, structure de référence, éclairage proche, composante indirecte venant des probes seulement

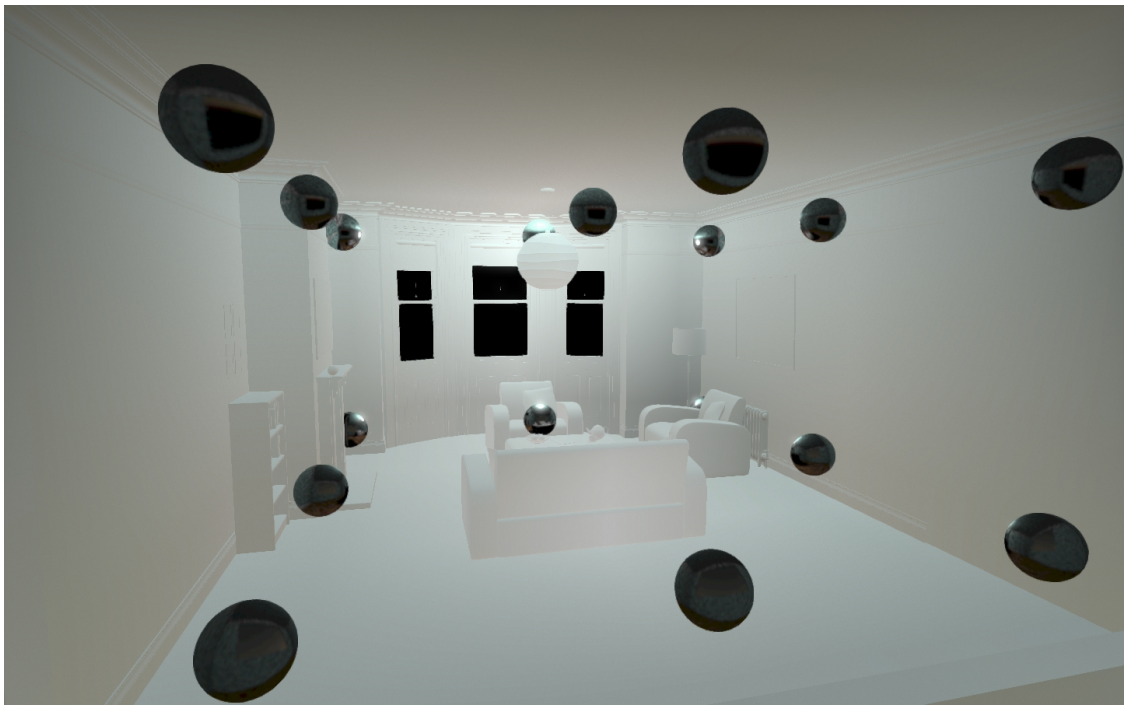


Figure 5.24 – Salon, structure de référence, éclairage proche, sans texture.

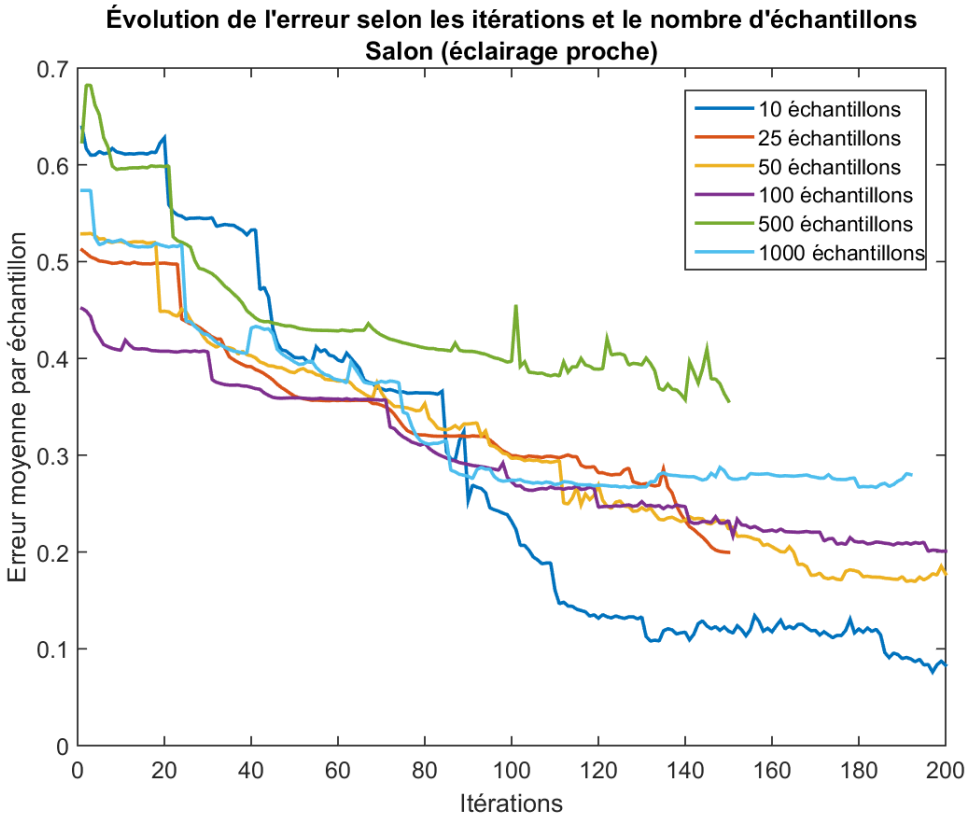


Figure 5.25 – Résultats de l'optimisation, Salon, éclairage proche.

Nombre d'échantillons	Erreur minimale	Erreur minimale moyenne	Nombre de probes
10	0.659	0.066	12
25	4.175	0.167	15
50	7.315	0.146	15
100	18.278	0.182	15
500	139	0.278	15
1000	266.54	0.266	8

Tableau 5.IX – Résultats du processus d'optimisation selon le nombre d'échantillons, Salon, éclairage proche.

Nombre d'échantillons	Itération	Nombre de probes
10	43	3
25	24	2
50	20	2
100	74	3
500	146	6
1000	62	5

Tableau 5.X – Itération et nombre de probes correspondant à partir de laquelle l'erreur des probes optimisés est inférieure à l'erreur de référence, Salon, éclairage proche.

CHAPITRE 6

DISCUSSION

6.1 Analyse de notre méthode

À la lumière de ces résultats, nous ne pouvons pas en toute bonne foi dire que notre objectif a véritablement été atteint. Nous avons tout de même réussi à élaborer, implémenter puis appliquer une méthode pour générer une structure de probes automatiquement au sein d'une scène, sans intervention humaine. Il est encourageant que les suppositions que nous avons faites se soient avérées vraies : en gardant les déplacements petits, et avec le bon traitement mathématique, l'optimisation par la réduction de l'erreur sur les coefficients SH fonctionne et nous arrivons à faire descendre itérativement notre valeur d'erreur.

Cependant, il est faux de dire que nous avons véritablement atteint notre objectif. Plusieurs critères que nous avons jugés comme nécessaires à l'obtention d'un résultat satisfaisant n'ont pas été remplis. Nous traiterons, dans cette section, plus en détail des problèmes liés à notre solution ; ils sont nombreux, et il faudrait encore beaucoup de travail pour tous les régler.

Notre choix de fonction d'erreur n'est au final, pas bien corrélé avec la qualité perçue du résultat ; la valeur qu'elle mesure ne semble pas être un bon indicateur de qualité du *shading*. Nous n'avons pas pu garantir la consistence de nos résultats, alors que des valeurs d'erreur similaires donnent des ensembles de probes, et donc des *shading* finaux très différents. De plus, nous avons testé notre méthode dans un nombre très limité de scènes, et n'avons pu obtenir signe de réussite que dans l'une d'entre elles ; cependant, pas pour tous les nombres d'échantillons de référence.

Notre choix de fonction d'interpolation n'était pas satisfaisant ; celle-ci peut être appropriée pour de petites scènes telles que le salon, mais n'est pas vraiment sensée dans une plus grosse scène telle que Sponza. De plus, nous n'avons pas testé les impacts qu'un choix différent d'exposant pour les valeurs de poids d'interpolation pourrait avoir.

Nous n'avons pas considéré le terme de visibilité en permettant à tous les probes d'avoir un impact sur toutes les surfaces dans la scène. Il est évident qu'un système utilisé en situation professionnelle ne pourrait accepter une telle faille.

Nos probes ne capturent que le *shading* diffus, et nous avons complètement négligé le *shading* spéculaire. De plus, comme nos probes ont été générés avec *Mitsuba* mais que notre moteur de rendu en temps réel est *G3D*, l'information contenue dans les probes ne correspond pas tout à fait à l'illumination directe produite par le moteur. L'utilisation de *Mitsuba* a ajouté une couche de complexité au système, l'information devant circuler entre deux programmes différents. Ceci a non seulement rendu notre programme difficile à *débugger* et à utiliser, mais a aussi fait que son exécution était beaucoup plus lente que ce que qui serait acceptable en contexte professionnel.

Ces critiques mettent en évidence le fait que notre méthode n'est que le début d'une réponse au problème que nous avons posé. Nos résultats suggèrent qu'il est envisageable d'éventuellement atteindre une solution universelle, rapide, consistente, simple d'utilisation et visuellement convaincante, mais y parvenir requiérerait de se pencher minutieusement sur les points soulevés. Au long de ce chapitre, nous examinerons plus en détails sur les points soulevés précédemment et tenterons de tracer des pistes de solutions.

6.2 Qualité du *shading* final

Il est clair que l'obtention d'une valeur plus basse à la fonction d'erreur telle que nous l'avons définie ne soit pas garante d'un meilleur rendu, comme on peut le voir sur nos résultats. L'approximation produite par nos structures de probes est inconsistante sur les surfaces ; on y retrouve des variations de hautes fréquences qui ne sont absolument pas plausibles, et des détails sont tout simplement éliminés. On peut le constater à la figure 6.1 ; l'éclairage varie de façon incohérente, et les interréflexions colorées des draperies qui étaient sur les images de référence sont complètement éliminées.

La scène du salon souffre de problèmes similaires. Que ce soit en prenant les probes correspondant à l'itération où l'erreur bat la référence (figure 6.2) où à une itération beaucoup plus avancée avec 15 probes (figure 6.3), le résultat visuel est chaotique et

Figure 6.1 – Irradiance interpolée dans Sponza, 2000 échantillons, itération 273.

diffère énormément de ce qu'on s'attendrait à voir.

Une fois que les textures sont appliquées sur les surfaces et qu'on passe d'une visualisation de l'irradiance incidente à la radiance sortante, beaucoup de détails sont éliminés, et les inconsistences sont moins évidentes. Ceci est une mince consolation, et une implémentation de production de notre algorithme ne pourrait accepter de tels problèmes. Il faudrait ajouter des contraintes à notre définition de l'erreur ou à notre méthode d'échantillonnage qui forcent l'approximation à produire des variations de moins hautes fréquences.

6.3 Interpolation

Le choix d'utiliser la pondération inverse à la distance (PID) pour interpoler entre tous les probes est un point faible de notre analyse. Ce choix était motivé par l'obligation de continuité de la fonction interpolante pour produire les dérivées nécessaires à l'optimisation. Les fonctions associant les poids aux probes pour les interpolations tri-

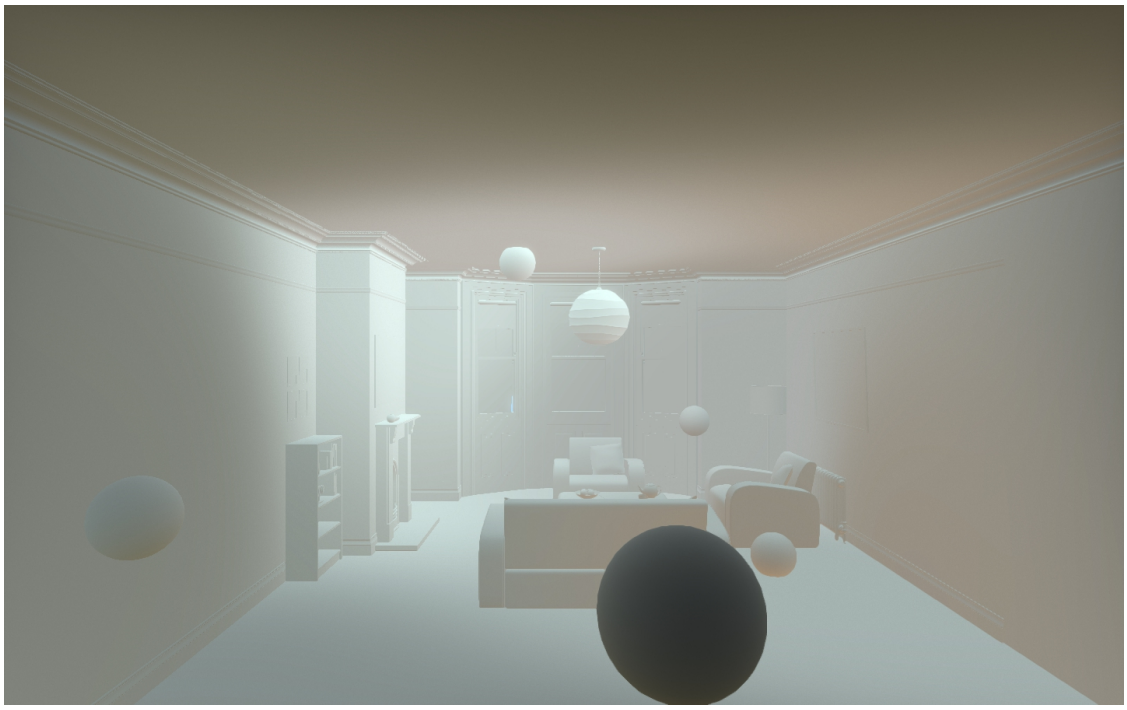


Figure 6.2 – Irradiance interpolée dans Sponza, 1000 échantillons, itération 74.

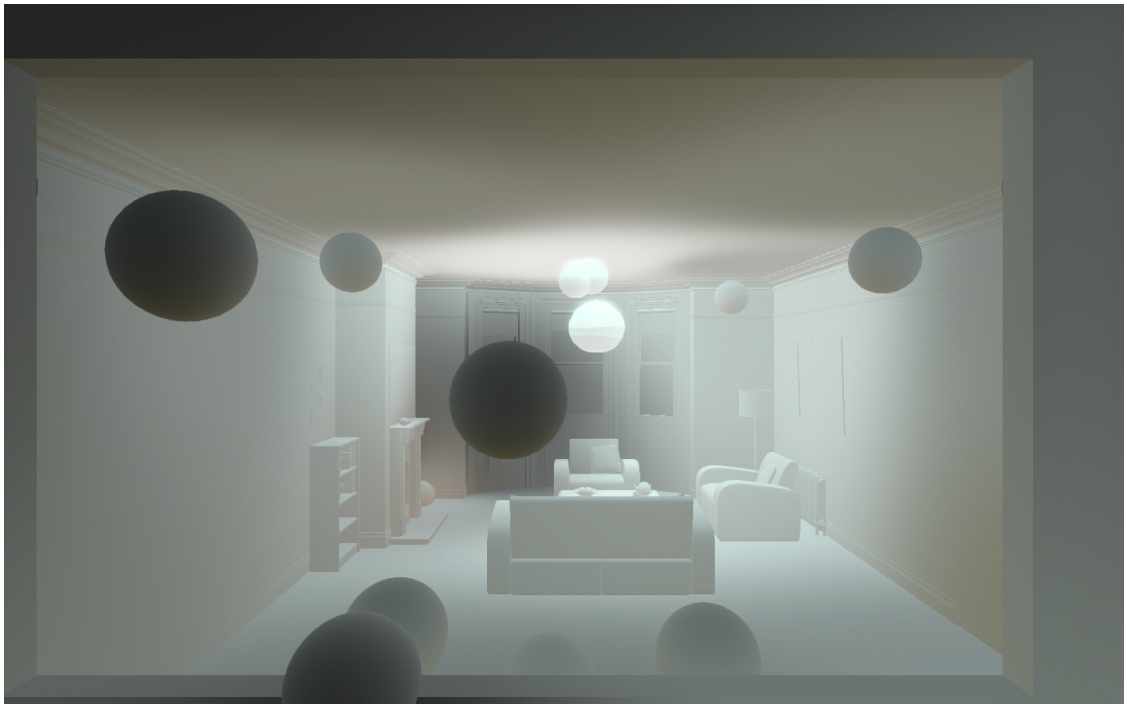


Figure 6.3 – Irradiance interpolée dans le salon, 1000 échantillons, itération 278.

linéaires et par maillage tétraédral présentent des discontinuités très abruptes selon si l'échantillon à interpoler se trouve dans la bonne cellule ; on ne peut donc les utiliser directement avec notre formulation. De plus, pour l'optimisation trilinéaire, les probes doivent bouger tous ensemble ; on ne pourrait trouver un vecteur de déplacement par probe. Une avenue pertinente serait d'optimiser des probes sur une grille régulière, et d'optimiser en faisant varier les dimensions de la grille et sa position initiale.

Il aurait peut-être été possible d'utiliser l'interpolation par maillage tétraédral en modifiant la valeur des poids pour enlever les discontinuités. Par exemple, plutôt que de tomber abruptement à 0 quand le point à interpoler sort de la cellule, les poids pourraient s'éteindre très vite de façon continue grâce à un facteur d'extinction dépendant de la distance du probe à la cellule. Nous n'avons fait aucune recherche en ce sens, mais ce pourrait être une piste intéressante ; les poids seraient continus et pourraient potentiellement être dérivés. La dérivée obtenue serait certainement très complexe.

Le problème principal de l'interpolation PID pour les probes, surtout l'utilisation que nous en avons faite, est qu'il n'est pas logique que chaque probe influence le résultat du *shading*. Le résultat de l'interpolation aux échantillons devrait-il vraiment dépendre de *tous* les probes dans la scène, même ceux situés très loin ? Il serait préférable que non. Le paramètre α régissant l'importance relative des probes situés près, un bon choix de α pourrait potentiellement régler ce problème gênant. Nous n'avons pas analysé l'impact de ce paramètre sur la reconstruction, et l'avons gardé fixé à 2.

6.4 Visibilité

Le vrai problème apporté dans notre choix de PID est que nous n'avons pas considéré la *visibilité* entre les probes et les échantillons. Si la ligne de visée entre un probe p à $\vec{p}_p$ et un échantillon s à $\vec{p}_s$ est coupée par un élément de géométrie, est-il vraiment sensé d'attribuer un poids non-nul à p ? La réponse, selon nous, est *peut-être*. Si les points sont dans la même pièce dans la scène, une partie de l'information à $\vec{p}_p$ pourrait être utile à p_s, mais il est difficile de généraliser.

Nous avons choisi d'ignorer la composante de visibilité lors du calcul des poids

puisqu'elle introduit, tout comme mentionné précédemment, des discontinuités dans la fonction de poids. Un probe qui passe derrière une surface occultant la visibilité à partir de p_s verrait son poids tomber immédiatement à 0, ce qui aurait un mauvais effet sur l'optimiseur.

Ceci n'invalide pas totalement l'interpolation PID ; ça ne vient que poser des conditions supplémentaires pour que son utilisation soit valide, soit celle que les probes associés à chaque échantillon soient *plausibles*. Ceci suggère une façon légèrement différente d'utiliser notre méthode que celle que nous avons présentée. Nous n'avons pas testé cette extension à la méthode, mais il est pertinent de la mentionner puisqu'elle pourrait faire l'objet de travail supplémentaire. Plutôt que de définir le domaine d'optimisation comme la scène en entier, il pourrait être intéressant de définir différents domaines au sein de la scène et d'appliquer la technique d'optimisation sur chacun d'entre eux. Par exemple, pour Sponza, la cour intérieure pourrait former un de ces domaines, et les paliers autour de la cour pourraient également en contenir plusieurs. Chacun de ces domaines aurait son propre ensemble d'échantillons d'entraînement et sa propre structure de probes ; ainsi on s'assurerait d'amoindrir l'effet de la non-localité des probes en excluant les probes hors de la zone à considérer au moment du *shading* d'un point.

Cette séparation de la scène en zones ayant chacun leur volume est, comme on a vu au chapitre 2, déjà une pratique courante dans les moteurs de rendu modernes. Il serait intéressant alors d'appliquer notre méthode à chacune de ces zones.

6.5 Échantillonnage

Un autre aspect intéressant apporté par la séparation en zones de la scène serait de localiser la fonction d'erreur. Présentement, l'erreur est répartie sur la scène entière ; si celle-ci était localisée à des zones discrètes elle pourrait potentiellement être un meilleur baromètre. Une implémentation de production pourrait également fournir des façons alternatives de générer des échantillons. Nous avons préféré les répartir de façon uniforme, mais il pourrait être intéressant de pouvoir spécifier la densité d'échantillons désirée en différentes régions de la scène. Ainsi, les artistes d'éclairage pourraient forcer l'op-

timiseur à produire un meilleur résultat aux endroits où ils jugent qu'il est important que la reconstruction soit bonne, en accordant moins d'importance à d'autres zones, par exemple celles où le joueur ne peut pas aller. On peut même imaginer générer les points d'échantillonnage importants en observant le comportement des joueurs. On pourrait mesurer lors de sessions de tests les emplacements les plus fréquentés par les joueurs et les désigner comme des zones d'échantillonnage importantes.

6.6 Surajustement

Lorsque la densité d'échantillonnage est faible, nous obtenons fréquemment des erreurs moyennes très faibles. Ceci reflète un phénomène qui peut se présenter dans tous les processus d'optimisation, soit celui du *surajustement*.

Le surajustement se produit quand l'optimiseur, plutôt que de capturer les tendances générales de variation d'une fonction, capture plutôt les tendances exactes de l'ensemble d'entraînement. L'optimiseur n'a alors pas à faire de "compromis" entre les échantillons, comme pour les ensembles d'entraînement plus grands ; il peut tout simplement apprendre les données d'échantillonnage et les représenter presque exactement.

Le problème est qu'un ensemble de probes surajusté ne sera valide que pour le petit nombre d'échantillons avec lequel il a été entraîné ; il ne sera pas bon pour reconstruire l'irradiance à d'autres endroits qu'à ceux de ces échantillons. À titre d'exemple, nous avons pris l'ensemble de probes créé avec 10 échantillons dans la boîte de Cornell, puis avons examiné sa performance avec 1000 échantillons. Le résultat est une augmentation de l'erreur moyenne d'environ 25% par rapport aux performances de l'ensemble de probes entraîné sur 1000 échantillons. Si le but est de reconstruire l'irradiance de façon uniforme dans la scène, il importe alors d'utiliser un nombre d'échantillons approprié, pas trop bas, afin d'éviter le surajustement.

6.7 Optimisations alternatives

Nous avons minimisé l'erreur sur la projection des coefficients SH en des points aléatoirement distribués de l'espace en modifiant la répartition spatiale des probes, mais

différentes avenues auraient pu être exploitées pour atteindre notre objectif.

Par exemple, la version initiale de notre optimiseur prenait ses échantillons non pas au sein de l'espace, mais plutôt sur les surfaces constituant la scène. Plutôt que de définir la cible comme un vecteur de coefficients SH, elle était plutôt définie comme l'ensemble des couleurs diffuses résultant de l'interpolation et de la reconstruction de l'irradiance. Les éléments de la matrice liée aux coefficients étaient donc constitués de sommes de coefficients sur toutes les bandes SH considérées, plutôt que de coefficients individuels. Nous avons préféré passer à des échantillons volumétriques puisque ceux-ci peuvent représenter les espaces entre les surfaces plutôt que seulement les surfaces, mais divers problèmes peuvent requérir diverses solutions ; il est concevable qu'il existe des problèmes où seule la reconstruction sur les surfaces soit d'intérêt, et cette méthode d'optimisation serait alors préférable.

Une autre avenue à explorer serait celle où plutôt qu'agir sur les positions des probes, on agit directement sur les coefficients qu'ils contiennent, sans les déplacer. Il serait alors possible d'appliquer l'optimisation sur des structures de probes pour lesquelles on ne connaît pas le gradient des poids d'interpolation, par exemple trilinéaires ou tétraédrales. Les coefficients ne correspondraient alors plus aux projections des "vraies" fonctions d'irradiance incidente aux emplacements des probes correspondants, mais plutôt à des fonctions intermédiaires n'ayant pas de sens physique autre qu'être optimisées pour l'interpolation.

Nous avons utilisé les coefficients SH, mais il existe d'autres bases pour représenter des fonctions définies sur une sphère. Par exemple, la compagnie Valve utilise une base spécialisée développée à l'interne [9] ; il pourrait être intéressant de voir si notre algorithme fonctionne avec une telle base et à quel point les résultats pourraient en être affectés.

CHAPITRE 7

CONCLUSION

Pour conclure, faisons un bref retour sur le travail que nous avons accompli. Nous avons développé, implémenté et appliqué une méthode basée sur la minimisation d'une fonction d'erreur objective pour le placement automatique de probes d'irradiance au sein d'une scène 3D.

Notre méthode se fonde sur l'observation que l'irradiance incidente est une fonction définie sur la sphère unitaire, continue et variant généralement lentement à petite échelle. Ainsi, l'expansion en série de Taylor au premier degré peut fournir une approximation de cette variation sur une courte distance. Nous utilisons cet outil pour approximer la variation d'une approximation de la fonction, reconstruite par interpolation à un point d'échantillonnage.

Nous avons développé les équations permettant d'exprimer cette variation, qui est une combinaison de deux termes indépendants. D'abord, on retrouve une contribution provenant du gradient des coefficients contenus dans les probes ; ceux-ci sont trouvés en échantillonnant la région autour de chaque probe puis en appliquant le théorème des différences finies. La seconde contribution provient des dérivées des poids associés à l'interpolation par rapport au déplacement de chaque probe dans la structure. La continuité étant une condition nécessaire au calcul des dérivées, nous avons utilisé une méthode d'interpolation PID.

Nous avons défini la fonction d'erreur comme la somme des carrés des différences entre les coefficients SH des fonctions d'irradiances échantillonnées et reconstruites. Ces différences écrites sous une forme vectorielle nous permettent d'exprimer le problème comme la résolution d'une équation matricielle de type $M \cdot \vec{x} = \vec{b}$, où M est une matrice associée aux poids d'interpolation et $\vec{b}$ le vecteur des variations à apporter aux coefficients reconstruits. Ce type de problème est bien étudié et des outils de résolution puissants sont abondamment disponibles. L'utilisation d'une implémentation de la méthode du gradient conjugué nous permet de trouver un vecteur $\vec{x}$ qui satisfait l'équation,

représentant les déplacements à apporter à chaque probe pour faire varier les coefficients dans la direction désirée.

Nous avons développé une méthode itérative de construction de structure de probes optimisée se basant sur ces principes. Cette méthode comporte deux phases en alternance : la première cherche un emplacement adéquat pour l'ajout d'un probe à la structure en construction, tandis que la seconde laisse relaxer la structure pour faire descendre le terme d'erreur global.

Nous avons appliqué notre méthode à des scènes réelles ; bien que la construction de structures de probes qui font baisser itérativement la fonction d'erreur soit une réussite, notre objectif d'obtenir une erreur inférieure à celle donnée par une structure de probes de référence se basant sur l'interpolation trilinéaire a eu un succès mitigé. Les résultats dans la scène du salon sont bons, alors que dans la boite de Cornell et Sponza à partir d'une cinquantaine d'échantillons nos probes ne font pas mieux que la référence.

Nous avons observé qu'une valeur d'erreur plus bas ne mène pas nécessairement à la perception d'un meilleur rendu ; nous avons découvert qu'il peut être meilleur de se fier sur une méthode plus subjective mais moins exacte pour l'interpolation (par moins exacte, nous voulons dire qui mène à un score d'erreur plus haut) que sur une méthode automatique plus exacte. En particulier, certains détails importants peuvent être omis, l'algorithme tentant de répartir l'erreur sur tous les points d'échantillonnage sans juger du résultat comme le ferait un humain.

L'interpolation par pondération à la distance inverse (PID) est un bon choix pour l'application des équations d'optimisation, mais un choix discutable pour ce qui est de l'interpolation entre probes ; celle-ci mène à des artéfacts dus au fait que tous les probes du domaine influencent chaque point à éclairer. Nous avons offert une piste de solution en ce sens : la même technique pourrait être appliquée à des sous-domaines de la scène, menant à des termes d'erreurs locaux qui pourraient être plus représentatifs que de tenter de tout caser sous la bannière d'un terme d'erreur global.

Il reste intéressant, selon nous, de baser la recherche en construction de structures de probes sur des données objectives. Nous avons déploré que les auteurs avant nous n'aient pas été assez rigoureux en ne considérant pas de mesure d'erreur objective, et bien que

nos résultats soient mitigés et qu'il faille résoudre plusieurs problèmes pour que notre méthode soit jugée satisfaisante, nous restons convaincus que de se baser sur ce type de mesure est la voie à suivre.

Il serait fantastique que ce type de méthode fasse son chemin jusque dans les moteurs de rendu professionnellement utilisés de nos jours. Le placement de probes est toujours un processus qui doit être fait à la main de façon plus ou moins ad-hoc, et les méthodes utilisées tendent à se baser sur la géométrie plutôt que les configurations lumineuses à représenter. Nous espérons que ce type de recherche soit un pas vers l'automatisation, qui utiliserait mieux le temps des artistes et aurait la capacité de produire des résultats plus exacts.

BIBLIOGRAPHIE

[1] James F. BLINN et Martin E. NEWELL. Texture and reflection in computer generated images. *Commun. ACM*, 19(10):542–547, octobre 1976. ISSN 0001-0782. URL http://doi.acm.org/10.1145/360349.360353.

[2] Paul BOURKE. Interpolation methods. *World Wide Web, http ://paulbourke.net/miscellaneous/interpolation/*, 1999.

[3] Gilles BRASSARD et Paul BRATLEY. *Fundamentals of Algorithmics*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996. ISBN 0-13-335068-1.

[4] Guedis CARDENAS et Morgan MCGUIRE. Cornell box model. [Included in G3D], 2017.

[5] Wikimedia Commons. Eight neighboring points enclosing the interpolation point. created by users skorkmaz, stannered, electrokid., 2007. URL https://commons.wikimedia.org/wiki/File:Enclosing_points.svg.

[6] Wikimedia Commons. An image rendered using path tracing, demonstrating notable features of the technique. created by user qutorial., 2016. URL https://commons.wikimedia.org/wiki/File:Path_tracing_001.png.

[7] Cyril CRASSIN. Voxel Cone Tracing and Sparse Voxel Octree for Real-time Global Illumination. Dans *GPU Tech Conference*, 2012. URL http://on-demand.gputechconf.com/gtc/2012/presentations/SB134-Voxel-Cone-Tracing-Octree-Real-Time-Illumination.pdf.

[8] Robert CUPISZ. Light Probe Interpolation Using Tetrahedral Tessellations. Dans *Game Developer's Conference*, 2012. URL http://www.gdcvault.com/play/1015312/Light-Probe-Interpolation-Using-Tetrahedral.

[9] Jason MITCHELL et al. Shading in Valve's Source Engine. Dans *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, pages 129–142, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6. URL http://doi.acm.org/10.1145/1185657.1185832.

[10] JAY-ARTIST et al. Cornell box scene. URL https ://t.co/yaFPeoWxUb, 2012.

[11] Matthäus G. CHAJDAS et Andreas WEIS et Rüdiger WESTERMANN. Assisted environment probe placement, Technische Universität München, Munich. 2011.

[12] Gaël GUENNEBAUD et Benoît JACOB et al. Eigen v3. http ://eigen.tuxfamily.org, 2010.

[13] Thomas ANNEN et Jan KAUTZ et Frédo DURAND et Hans-Peter SEIDEL. Spherical Harmonic Gradients for Mid-Range Illumination. Dans *Eurographics Workshop on Rendering*. The Eurographics Association, 2004. ISBN 3-905673-12-6.

[14] Brenda CHAPMAN et Mark ANDREWS. *Brave*. Pixar Animation Studios, 2012.

[15] Jack HOXLEY et Matt PETTINEO et Jason ZINK. *Practical Rendering and Computation with Direct3D 11*. A. K. Peters, Ltd., Natick, MA, USA, 1ère édition, 2011. ISBN 1568817207, 9781568817200.

[16] Mickael GILABERT et Nikolay STEFANOV. Deferred Radiance Transfer Volumes : Global Illumination in Far Cry 3. Dans *Game Developer's Conference*, 2012. URL http://www.gdcvault.com/play/1015326/Deferred-Radiance-Transfer-Volumes-Global.

[17] Cyril SOLER et Olivier HOEL et Frank ROCHET. A deferred shading pipeline for real-time indirect illumination. Dans *ACM SIGGRAPH 2010 Talks*, SIGGRAPH '10, pages 18 :1–18 :1, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0394-1. URL http://doi.acm.org/10.1145/1837026.1837049.

[18] Philip DUTRE et Paul HECKBERT et Vincent MA et al. Global illumination compendium. Department of Computer Science, Katholieke Universiteit Leuven, 2001.

[19] Gene GREGER et Peter SHIRLEY et al. The irradiance volume. *IEEE Computer Graphics and Applications*, 18(2):32–43, Mar 1998. ISSN 0272-1716.

[20] USC Institute for Creative Technologies. Radiance probe for Grace Cathedral, San Francisco, California, 2006. URL `http://gl.ict.usc.edu/Data/HighResProbes/`.

[21] Shawn HARGREAVES. Deferred Shading. Dans *Game Developer's Conference*, 2004. URL `http://www.shawnhargreaves.com/DeferredShading.pdf`.

[22] Naty HOFFMAN. Background : Physically-Based Shading. Dans *ACM SIGGRAPH Physically-Based Shading Models in Film and Game Production Course*. ACM, 2010. URL `http://renderwonk.com/publications/s2010-shading-course/hoffman/s2010_physically_based_shading_hoffman_a_notes.pdf`.

[23] JT HOOKER. Volumetric Global Illumination At Treyarch. Dans *ACM SIGGRAPH Advances in Real-Time Rendering Course*. ACM, 2016. URL `http://bit.ly/2pxNqwl`.

[24] James T. KAJIYA. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4): 143–150, août 1986. ISSN 0097-8930. URL `http://doi.acm.org/10.1145/15886.15902`.

[25] Remedy Entertainment Ltd. Quantum Break. [Blu-ray], 2016.

[26] Morgan MCGUIRE et Crytek. Sponza model. [Included in G3D], 2011.

[27] Gene MILLER et Robert HOFFMAN. Illumination and Reflection Maps : Simulated Objects in Simulated and Real Environments. Dans *ACM SIG-*

GRAPH Course Notes for Advances Computer Graphics Animation. ACM, 1984. URL http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.9.4747&rep=rep1&type=pdf.

[28] Torsten MÖLLER. Notes de cours : Visualization (cmpt 467/767), Simon Fraser University, Burnaby, Canada, Septembre 2010. URL http://www.cs.sfu.ca/~torsten/Teaching/Cmpt467/.

[29] Derek NOWROUZEZAHRAI. Notes de cours : Sujets en infographie (IFT6095), Université de Montréal, Hiver 2014.

[30] Matt PHARR et Greg HUMPHREYS. *Physically Based Rendering, Second Edition : From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2ème édition, 2010. ISBN 0123750792, 9780123750792.

[31] Ravi RAMAMOORTHI et Pat HANRAHAN. An efficient representation for irradiance environment maps. Dans *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 497–500, New York, NY, USA, 2001. ACM. ISBN 1-58113-374-X. URL http://doi.acm.org/10.1145/383259.383317.

[32] Yossi RUBNER et Carlo TOMASI et Leonidas J. GUIBAS. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, Nov 2000. ISSN 1573-1405. URL https://doi.org/10.1023/A:1026543900054.

[33] Ari SILVENNOINEN et Ville TIMONEN. Multi-Scale Global Illumination in Quantum Break. Dans *ACM SIGGRAPH Advances in Real-Time Rendering Course*. ACM, 2015. URL http://wili.cc/research/quantum_break/.

[34] Peter-Pike SLOAN. Stupid spherical harmonics (SH) tricks. Dans *Game Developer's Conference (San Francisco)*, 2008. URL http://www.ppsloan.org/publications/StupidSH36.pdf.

[35] Petras SUKYS. Light probe cloud generation for games, Department of Computer Science and Engineering, Chalmers University of Technology. Mémoire de maîtrise, 2014.

[36] Natalya TATARCHUK. Irradiance volumes for games. Dans *Game Developer's Conference (Europe)*, 2005. URL `https://developer.amd.com/wordpress/media/2012/10/Tatarchuk_Irradiance_Volumes.pdf`.

[37] Michal VALIENT. Taking Killzone Shadow Fall Image Quality into the Next Generation. Dans *Game Developer's Conference*, 2014. URL `http://www.gdcvault.com/play/1020770/Taking-Killzone-Shadow-Fall-Image`.

[38] Joseph VOLPE. Disney rendered its new animated film on a 55,000-core supercomputer. *World Wide Web, https ://www.engadget.com/2014/10/18/disney-big-hero-6/*, 2014.

[39] Jarosz WOJCIECH. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. Thèse de doctorat, UC San Diego, La Jolla, CA, USA, septembre 2008.